

** DE 6502 KENNERS ** — EEN CLUB VOOR 6XXXX GEBRUIKERS

De vereniging heeft leden in Nederland, België, Duitsland, Frankrijk, Spanje, Portugal, Amerika, Zambia, Denemarken. Het doel van de vereniging is: het bevorderen van de kennisuitwisseling tussen gebruikers van 6XXXX-computers, als COMODORE-64, APPLE, CHE-1, PEARCOM, AIM-65, SYM, PET, BBC ATARI, VIC-20, BASIS 108, PROTON-computers, ITT-2020, OSI, ACC 8000, ACDM ELECTRON, SYSTEM 65, PC-100, PALLAS, MINTA FORMOSA, ORIC-1, STARLIGHT, CV-777, ESTATE III, SBC65/68, KIM, MCS, KEMPAC SYSTEM-4, Elektuur-computers (JUNIOR, en de OCTOPUS), LASER, maar ook 6800, 6809 en 68000-computers. De kennisuitwisseling wordt o.a. gerealiseerd door 6 maal per jaar DE 6502 KENNER te publiceren, door het houden van landelijke clubbijeenkomsten, door het instandhouden van een cassette-bibliotheek en door het verlenen van paperware-service. Regionale bijeenkomsten worden door de leden georganiseerd.

Verschijningsdata
DE 6502 KENNER 1985

derde zaterdag van
februari, april, juni,
augustus, oktober, december.

Inlichtingen over de regio-
bijeenkomsten!

Berard van Roekel
Van der Palastraat 11 - C
3135 LK Vlaardingen
Tel.: 010 - 351101

De vereniging is volledig onafhankelijk, is statutair opgericht en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305.

Voorzitter!
Rinus Vleesch-Dubois
Fl. Nightingalestraat 212
2037 NG Haarlem
Tel.: 023 - 330993

Penningmeester:
John F. van Sprang
Tulp 71
2925 EW Kripen/IJssel.
Tel.: 01807 - 20589

Leden:
Adri Mankel (05490 - 51151) Hardware/software/DOS65
Erwin Visschedijk (05490 - 71416) Hardware/software/DOS65
Bert van Opbroek (01729 - 8636)
Nico de Vries (010 - 502239) Hardware/software/PET
Erevoorzitter: Siep de Vries
Ereleden: Mw. H. de Vries - Van der Winden
Anton Mueller
Lidmaatschap: Hfl. 45,- per kalenderjaar, postrekening 3757649 t.n.v. Penningmeester KIM Gebruikers Club Ned., Kripen/IJssel.

Advertenties: Tarieven op aanvraag bij de redactie.

** DE 6502 KENNER ** — EEN BLAD VOOR 6XXXX GEBRUIKERS

DE 6502 KENNER is een uitgave van de KIM Gebruikers Club Nederland. Het blad wordt verstrekt aan leden van de club. DE 6502 KENNER wordt van kopij voorzien door leden van de club, bij de opmaak van een publikatie bijgestaan door de redactie. De inzendingen van programma's dienen voorzien te zijn van commentaar in de listings en zo mogelijk door een inleiding voorafgegaan. Publikatie van een inzending betekent niet dat de redactie of het bestuur enige aansprakelijkheid aanvaardt voor de toepassing ervan. De inzendingen kunnen geschieden in assembly-source-listings, in Basic, in Basicode, Forth, Focal, Comal, Pascal, Fortran, Cobol, Logo Elan, etc. etc. De leden schrijven ook artikelen over de door hen ontwikkelde hardware en/of aanpassingen daarop. Zij schrijven tevens artikelen van algemene aard of reageren op publikaties van andere inzenders.

DE 6502 KENNER IS EEN BLAD VAN EN DOOR DE LEDEN

Micro-ADE Assembler/Disassembler/Editor is een produkt van Micro Ware Ltd., geschreven door Peter Jennings en bestemd voor alle 6502-computers. De KIM Gebruikers Club Ned. heeft de copyrights verworven nadat ons lid Sebo Woldringh de 4 K KIM-1 versie uitbreidde tot 8 K KIM-1 versie. Adri Mankel paste deze aan voor de JUNIOR. Willem L. van Pelt stelde een nieuwe 8 K source-listing voor de JUNIOR samen. De implementatie op andere systemen dan de KIM-1 en JUNIOR kan eenvoudig gebeuren door het aanpassen van de I/O-adressen, welke in de source-listing gemakkelijk te vinden zijn.

FATE Format-listor/cond. Assembler/Tape-utilities/Editor is de door ons lid Rob Banen geschreven source-listing van een 12 K universeel systeem voor de JUNIOR-computer aan de hand van het universele disk operating system van de fa. Proton Electronics te Naarden, nu geschikt voor werken met tapes. FATE wordt beschikbaar gesteld met toestemming van Proton.

In de edities van DE 6502 KENNER worden regelmatig mededelingen gedaan over de door de club georganiseerde bijeenkomsten. Ook worden bestuurlijke mededelingen gedaan, naast informatie over hetgeen in de handel te koop is. Leden die iets te koop hebben of iets zoeken kunnen dit in de edities van DE 6502 KENNER bekend maken. Ook worden brieven aan de redactie gepubliceerd, evenals specifieke vragen van leden. De edities worden samengesteld op basis van een groot aantal prioriteiten, welke door een redactievergadering worden gehanteerd. Deze vergadering bestaat uit de vaste medewerkers zoals in de colofon vermeld. Het aantal inzendingen is groter dan in een enkele editie van minimaal 48 pagina's is te verwerken. Hierdoor kan het voorkomen dat een inzending eerst na enige tijd kan worden gepubliceerd.

DE CLUB HEEFT BEHOEFTE AAN MEER LEDEN. WIJ WILLEN MEER AAN KUNNEN BIJEN DAN NU AL HET GEVAL IS. WERF DAAROM EEN LID!

WILT U EEN PRIJSLIJST? STUUR EEN GEFRANKEERDE ENVELOP AAN HET REDAKTIE-ADRES.

Een onafhankelijke jury kent jaarlijks een aantal aanmoedigingspremies toe aan auteurs van gepubliceerde artikelen in DE 6502 KENNER.

De 6502 KENNER is een uitgave van de KIM gebruikers Club Nederland.

Adres voor het inzenden van en reacties op artikelen voor DE 6502 KENNER:
Willem L. van Pelt
Jacob Jordaensstraat 15
2923 CK Krimpden a/IJssel
Tel.: 01807 - 19881

Vaste medewerkers:
Willem L. van Pelt
Gerard van Roekel
Frans Smeehuijzen

Freelance medewerkers:
Rob Banen
Fred Behringer, (Germany)
Fridus Jonkman
Gert Klein
Roger Langeveld
Fernando Lopes, (Portugal)
Frank Manshande
Gert van Opbroek
Ruud Uphoff
Frans Verberkt
Herman Zondag

Gehele of gedeeltelijke overname van de inhoud van DE 6502 KENNER zonder toestemming van het bestuur is verboden. Toepassing van gepubliceerde programma's, hardware etc. is alleen toegestaan voor persoonlijk gebruik.

DE 6502 KENNER verschijnt 6 x per jaar en heeft een oplage van 500 exemplaren.

Copyright (C) 1986 KIM Gebruikers Club Nederland.

De voorpagina is de DOS65-controllerkaart. ontwikkeld door Ad Brouwer.
CAD/CAM: E. Visschedijk.
I.s.m. : A. Hankel
Fotogr.: Fr. Visschedijk.

In verband met auteurswetgeving en andere maatregelen op het gebied van bescherming van software kan de redactie geen aansprakelijkheid aanvaarden voor inzendingen. Inzendingen dienen afkomstig te zijn van de inzender, tenzij anders aangegeven.

INHOUDSOPGAVE DE 6502 KENNER NR. 44 JUNI 1986

1. Alternatieve Break Routine t.b.v. CPU-kaart ... Frans Smeehuijzen	2.
2. Van de redactie	3.
3. APPLE: Catalog Header ... Frank Manshande	3.
4. Elektor's OCTOPUS/EC65 computer: A Data-Buffer as an Afterthought ... R.T. Overakker	4.
5. DOS65: DOS65 Corner ... Coen Kleipool, France	5.
6. Elektor's OCTOPUS/EC65 computer: Modified Diskette Copier Version 2.2 part 2 ... Wolfgang Tietsch	7.
7. JUNIOR: Interfacekaart en gewijzigde VDU-kaart JUNIOR: Interfacecard and modified VDU-card (hardware schemes) ... Pieter de Visser	18.
8. JUNIOR: POSVAL SCHAAKMONITOR part 1 JUNIOR: POSVAL CHESSMONITOR for Elektor's JUNIOR with hexdisplay ... Frans Raaijmakers	21.
9. OCTOPUS: Printer Initialisation for EC65 ... Leif Rasmussen, Denmark	29.
10. BASIC: Competitiestanden (Handbal) part 3 Printroutine voor teletype 110 baud ... Gerard Keet	31.
11. C-16: Tokenization of the BASIC Instructions ... Fred Behringer, Muenchen	34.
12. BASIC: OTHELLO game in Basic	36.
13. JUNIOR with Proton's Senior Monitor Modified Format-Lister for Senior Monitor ... Rob Banen	38.
14. 6845: De 6845 geprogrammeerd 6845: Programming the 6845 (see Elektor Holland, Oct. 1984) ... H. Feijen	45.
15. Elektor's VDU-card: Insert - Delete character routines t.b.v. de VDU-kaart ... Frans Smeehuijzen	48.
16. DIVERSEN: OCTOPUS/EC65: Problems with power-on-reset ... Siegfried Losensky, Germany ATARI: Sprites on Atari OCTOPUS/EC65: Patch Real Time Clock (see Elektor Holland, April '86) APPLE: Bug in Graphics in Applesoft, DE 6502 KENNER Dec. '85 C-64: Converting Tokenized Basic into regular Basic for C-64 C-16: Supertape on the C-16 ... Fred Behringer, Germany APPLE: Verandering van Uw Basicode-2 routines ... Frans Verberkt 6502/6510: 6502/6510 verschillen VDU-kaart: Een paar problemen ... Stefan Sperling, Belgium OCTOPUS/EC65: SAMSON tips ... Leif Rasmussen, Denmark News from the DRAM Front ... Fred Behringer, Germany	6. 21. 21. 22. 22. 22. 23. 24. 24. 29. 35.

BREAK ROUTINE

PROTON 650X ASSEMBLER V4.4 PAGE: 0001

```

0001 0000      .TIT 'BREAK ROUTINE'
0002 0000      ;
0003 0000      .OPT GEN
0004 0000      ;
0005 0000      ;*****
0006 0000      *
0007 0000      * ALTERNATIEVE BREAK ROUTINE T.B.V. CPU-KAART *
0008 0000      *
0009 0000      ;*****
0010 0000
0011 0000      AUTEUR: F.J.M. SMEEHUIJZEN
0012 0000      LIPPEDAL 19
0013 0000      2904 CL CAPELLE AAN DEN IJSSEL
0014 0000      TEL: 010-512507
0015 0000
0016 0000      ; OM EEN BASIC PROGRAMMA- OF EEN DISASSEMBLER LISTING
0017 0000      ; OP HET BEELDSCHERM OF EEN PRINTER EVEN TE KUNNEN
0018 0000      ; ONDERBREKEN WORDT IN DE REGEL GEBRUIK GEMAAKT VAN
0019 0000      ; DE 'BREAK' TOETS.
0020 0000      ; OP DE PARALLEL-AANSLUITING VAN DE VIA 6522 OP DE
0021 0000      ; CPU-KAART VAN ELEKTUUR IS NIET ECHT EEN VOORZIENING
0022 0000      ; BETROFFEN OM EEN BREAK-TOETS AAN TE SLUITEN.
0023 0000      ; TEVENS BEZIT HET DOOR MIJ GEBRUIKTE TOETSENBOORD
0024 0000      ; OOK GEEN ECHTE 'BREAK' TOETS.
0025 0000      ; VERDER HEEFT DE PROTON MONITOR EEN 'KEYBOARD INTERRUPT
0026 0000      ; ROUTINE' WELKE DOOR EEN BREAK WORDT AANGESPROKEN EN
0027 0000      ; VERVOLGENS DE MOGELIJKHEID BIEDT OM ENERZIJD DOOR
0028 0000      ; HET INTOETSEN VAN EEN WILLEKEURIGE TOETS DE VERWERKING
0029 0000      ; VOORT TE ZETTEN EN ANDERZIJD DOOR HET INTOETSEN VAN
0030 0000      ; EEN ESCAPE TERUG TE SPRINGEN NAAR DE MONITOR.
0031 0000      ; HET PROBLEEM VAN HET ONTBREKEN VAN EEN 'BREAK' TOETS
0032 0000      ; HEB IK ALS VOLGT OPGELOST:
0033 0000
0034 0000      ; DOOR HET GELIJKTIJDIG INTOETSEN VAN DE CONTROL+@
0035 0000      ; WORDT EEN ASCII 00 GEGENEREERD.
0036 0000      ; DE MONITOR ROUTINE RCHEK KIJKT NU NAAR EEN NULL CHARACTER.
0037 0000
0038 0000      VAPAD    = $1001      ; PORT A DATA REGISTER
0039 0000      GETCH    = $EA13      ; MONITOR GETKEY ROUTINE
0040 0000      COMIN    = $E19F      ; WARM RESTART OF MONITOR
0041 0000
0042 0000      ;
0043 0000      ;
0044 E9DE      ; *** KEYBOARD INTERRUPT ROUTINE ***
0045 E9DE
0046 E9DE      AD0110 RCHEK    LDA VAPAD      ; GET PORT A DATA
0047 E9E1      297F          AND #$7F        ; FILTER ASCII CHARACTER
0048 E9E3      D007          BNE RCHEK1
0049 E9E5      2013EA        JSR GETCH        ; WAIT FOR A KEY
0050 E9E8      C91B          CMP #$1B        ; WAS IT ESCAPE?
0051 E9EA      F00B          BEQ READ2        ; RETURN TO MONITOR
0052 E9EC      60           RCHEK1    RTS      ; IF NOT END OF INTERRUPTION
0053 E9ED
0054 E9ED      ; *** GET CHARACTER TO ACCU, ESCAPE POSSIBLE ***
0055 E9ED
0056 E9ED      2013EA READ    JSR GETCH        ; WAIT FOR A KEY
0057 E9F0      C91B          CMP #$1B        ; WAS IT ESCAPE?
0058 E9F2      D003          BNE READ2
0059 E9F4      4C9FE1 READ1   JMP COMIN      ; ON ESCAPE TO MONITOR
0060 E9F7      60           READ2    RTS
0061 E9F8
0062 E9F9      ;
      .END

```

ERRORS: 0000

(0000)

VAN DE REDAKTIE

Het werk op de redaktiekamer gaat gestaag voort en laat zich door geen koninginndag, 1 mei-viering, bevrijdingsdag, hemelvaartsdag of verplichte arbeidstijdverkorting verhinderen om door te gaan, of het moet zijn dat de beschikbare apparatuur het begeeft. Voor een buitenstaander moet het haast wel lijken of het hier om een tweemaandelijks cyclus gaat die op den duur dodelijk vervelend dreigt te worden. En hij heeft gelijk. Als er tenminste aan een voorwaarde voldaan wordt: er moet gewoon nooit iets opwindends voorkomen. Dat is nu juist wat mij zo ongelofelijk aan deze kant van onze hobby bindt. Er gebeurt iedere dag wel iets waarvoor elke idealist en freak als ik best wel rode koontjes bij kan krijgen. Neem nou het geval dat er op een vrijdag ineens een editie van mei 1986 van Elektor Electronics bij mij in de bus glijdt. Ach, ik krijg wel meer toegezonden zo af en toe. Maar toch, ik ben nieuwsgierig of er dezelfde artikelen in staan als in Elektor in Nederland. Het zal wel, want zij publiceren de ons inmiddels heel bekende printer-buffer. Dan valt ineens m'n oog op een kop "CALLING ALL 6502 USERS". Dat ben ik, denk ik nog. En plots zie ik mijn naam en adres. Verhip, die tekst, wat komt die me bekend voor! Geen wonder, ik schreef het zelf, maanden geleden al. Informatie over onze internationale club en onze activiteiten. Een halve pagina maar liefst, met daarboven "advertisement". Elektor Electronics komt uit in de UK, USA en in Ierland. Enfin, vanaf dat moment begin ik te gloeien. Jaminee, stel je voor, ik

moet me prepareren, erop voorbereiden, maatregelen nemen, alles in orde brengen, er komen natuurlijk postzakken vol reacties. Over de eerste koorts heen denk ik: als de donder alle redakties in en buiten Europa aanschrijven. We moeten zorgen dat zij dit voorbeeld gaan volgen. Want, wij hebben behoefte aan meer leden en we ondersteunen met onze uitwisseling van kennis hun doel immers ook. Als wij de leden weten te boeien met onze artikelen, dan zijn er veel mensen onder hen die hun eurokaarten gebruiken en als er veel mensen zijn die hun eurokaarten gebruiken, dan bieden wij hen weer een interessant dak boven het hoofd. In Engeland hebben de redactieleden van EE het begrepen, in Nederland en Duitsland was onze naam al flink genoemd, maar we zijn er nog lang niet.

Maar ja, wat doe je nou met zo'n probleem dat die mensen geen spaan Nederlands kunnen lezen, terwijl wij over het algemeen toch best in staat blijken het Engels te begrijpen? Nou, dan leg je dat de leden voor. En wat zie je? In minder dan geen tijd hebben zich een aantal leden gemeld die bereid blijken alle noodzakelijke teksten te vertalen van het Nederlands naar het Engels. Er waren allang een aantal leden die hun source-listings in het Engels commentarieerden of gehele artikelen in die vorm schreven, maar nu gaat het om materiaal van anderen, al dan niet reeds gepubliceerd. Ook hier kan ik best nog wel hulp gebruiken, maar ik wilde maar even kwijt, dat onze club weer eens laat zien dat het ook andere dingen aankan dan alleen aan het toetsenbord of aan de soldeerbout hangen.

Willem L. van Pelt.

```

1 REM CATALOG HEADER
2 REM DOOR
3 REM FRANK MANSHANDE
4 REM -----
5 REM
6 FOR RWTS = 896 TO 926
7 READ BYTE
8 POKE RWTS, BYTE
9 NEXT
10 REM
11 REM RWTS ROUTINE ZIT NU
12 REM IN HET GEHEUGEN
13 REM
14 DATA 169,3,160,138,32,217
15 DATA 3,96,0,0,1,96,1,0,0
16 DATA 0,155,3,0,128,0,0,1
17 DATA 0,0,96,1,0,1,239,216
18 REM
19 REM DIT WAREN DE DATA'S VAN
20 REM DE RWTS ROUTINE
21 REM
22 REM VARIABELEN :
23 REM
24 TR = 2:SE = 2
25 LEES = 1:SCHRIJF = 2
26 RET$ = CHR$(13)
27 ESC$ = CHR$(27)
28 H$ = CHR$(8)
29 RWTS = 896
30 REM
31 REM SCHERM OPMAAK
32 REM
33 SPEED= 255: NORMAL
34 TEXT : HOME :H = 5
35 PRINT "CATALOG HEADER DOOR FRANK MANSHANDE"
36 PRINT "-----"

```

```

370 VTAB 5
380 PRINT "DOE DE TE VERANDEREN DISKETTE IN DRIVE"
390 PRINT "1 EN DRUK DAN OP EEN TOETS...";
400 GET T$
410 IF T$ = ESC$ THEN 1150
420 POKE 910,TR: POKE 911,SE
430 POKE 918,LEES
440 CALL RWTS
450 REM
460 REM NU IS GEPROBEERD DE
470 REM BENODIGDE SECTOR TE
480 REM LADEN. WE KIJKEN TOCH
490 REM NOG VOOR DE ZEKERHEID
500 REM OF ER EEN FOUT WAS
510 REM
520 ERR = PEEK(919)
530 IF ERR = 16 THEN A$ = "WRITE PROTECTED"
540 IF ERR = 64 THEN A$ = "I/O ERROR"
550 IF ERR = 128 THEN A$ = "READ ERROR"
560 VTAB 20: HTAB 20 - LEN(A$)
570 FLASH : PRINT A$
580 NORMAL
590 IF LEN(A$) > 0 THEN A$ = "": GET T$: GOTO 340
600 REM
610 REM ER IS GEEN FOUT, DUS
620 REM GAAN WE DOOR MET HET
630 REM PROGRAMMA
640 REM
650 REM NU WORDT DE NIEUWE
660 REM TEKST GEVRAAGD EN DIE
670 REM MAG NIET MEER DAN 11
680 REM LETTERS ZIJN
690 REM
700 VTAB 5: HTAB 1: CALL - 958
710 PRINT "VOER DE TEKST IN (MAX. 11 LETTERS)"
720 PRINT : PRINT "==" ";IN$ = ""

```

```

730 HTAB H: GET T$
740 IF T$ = ESC$ THEN 340
750 IF T$ = RET$ THEN J = LEN (IN$): GOTO 810
755 IF T$ = H$ AND LEN (IN$) = 1 THEN IN$ = "":H = 5: GOTO 700
760 IF T$ = H$ AND LEN (IN$) > 1 THEN IN$ = LEFT$ (IN$, LEN (IN$) - 1):H = H - 1: VTAB 7: HTAB H: PRINT " ": HTAB H: GOTO 730
770 IF ASC (T$) < 32 THEN 730
780 IN$ = IN$ + T$: PRINT T$:
790 IF LEN (IN$) > 11 THEN H = 5: GOTO 700
800 H = H + 1: GOTO 730
810 REM
820 REM POKE NU DE TEKST IN DE
830 REM SECTOR BUFFER
840 REM
850 IF J = 0 THEN 880
860 FOR I = 13 - J TO 12
870 POKE 32942 + I, ASC ( MID$ (IN$,J,1)) + 128:J = J - 1: NEXT
880 FOR I = 32943 TO 32954 - LEN (IN$)
890 POKE I,160
900 NEXT
910 REM
920 REM WE GAAN NU DE SECTOR
930 REM WEER TERUG SCHRIJVEN
940 REM
950 POKE 910,TR: POKE 911,SE
960 POKE 918,SCHRIJF
970 CALL RWT$
980 REM
990 REM WE KIJKEN TOCH NOG
1000 REM VOOR DE ZEKERHEID OF
1010 REM ER EEN FOUT IS
1020 REM
1030 ERR = PEEK (919)
1040 IF ERR = 16 THEN A$ = "WRITE PROTECTED"
1050 IF ERR = 64 THEN A$ = "I/O ERROR"
1060 VTAB 20: HTAB 20 - LEN (A$)
1070 FLASH : PRINT A$
1080 NORMAL
1090 IF LEN (A$) > 0 THEN A$ = "": GET T$: GOTO 340
1100 REM
1110 REM ER WAS GEEN FOUT, DAN
1120 REM KUNNEN WE DUS HET
1130 REM PROGRAMMA VERLATEN
1140 REM
1150 HOME : END

```

3. Now load your bufferless program(PR).
Set Memory-pointer to free space in your memory and subsequently list the program into your memory.

DISK!"LO PR"

DISK!"ME 8000,8000"

LIST#5

PRINT#5,"POKE 893,1"

By these commands your program is invisibly copied in memory from \$8000 onward as an ASCII-file, followed by a POKE-command (see later on).

4. Next, load the aid-program; again set Memory-pointer and design your memory an I/O device (DSI-MANUAL, p.54)

DISK!"LO DUMMY"

DISK!"ME 8000,8000"

DISK!"IO 10,01"

Your bufferless program is now "typed" into the buffered program, followed by a listing on screen. Notice the POKE-command at the end of the listing. This command enables your keyboard. If your program hasn't got a linenumber 10, then you have to delete linenumber 10 originating from the aid-program.

5. DISK!"PU BUFFER"

saves your now well BUffered Program.

A DATA-BUFFER AS AN AFTERTHOUGHT.

System: Elektor's OCTOPUS-computer
Author: R.T. Overakker

Yes, I did type a 20 K program forgetting to create a buffer beforehand.
J.D.J. Derksen, a member of our club, helped me out neatly. This was done by using Memory as an I/O device:

1. Use option 7 of BEXEC*. Create buffer(s) as desired and type an aid-program:

10 REM DUMMY

2. Put this exciting program on disk. Later on you merge this buffered program with your bufferless one.
DISK!"PU DUMMY"

(NIEUWE) BOEKEN

BOUW ZELF EEN EXPERTSYSTEEM IN BASIC.
Chris Naylor/vert: A.M. Korff
1986/ca. 250 pag./f 45,=
ISBN 90 6233 167 X

WERKEN MET BESTANDEN OP DE COMMODORE-64.
G. Fisher/L. Finkel/J.R. Brown
vert: Jan Kuis
1986/ca. 480 pag./f 78,=
ISBN 90 6233 186 6

PROGRAMMEERCURSUS APPLESOFTBASIC.
Nok van Veen/Ad van Delft
1986/ca. 280 pag./f 45,=
ISBN 90 6233 174 2

6502 ASSEMBLEERTAAL EN MACHINECODE VOOR BEGINNERS
A.P. Stephenson
1984/ 206 pag./f 37,50
ISBN 90 6233 123 8

```

*****
*                                     *
*   D O S   6 5   C O R N E R   *
*                                     *
*****

```

Author: drs Conrad H. Kleipool, Val de Perier, F-83310 Cogolin, France.
tel: (33) 94.54.43.82.

NEW: DOS-65 VERSION 2.

Let's start this Dos65-corner with some good news. The Brouwer-Hankel-Visschedyk team has not been wasting their time since they brought out Dos65. They have just announced an upgraded version which has some exciting new features:

- 80 instead of 40 tracks;
- Directory divided in 7 subdirectories;
- Memory extension to accomodate a virtual disk up to 1 Mbyte;
- Extra DD formatting writing 18 sectors per track instead of 16. This results in 720 Kbytes formatted for an 80-tracks floppy.
- Parameters can be incorporated in command files.

Some of these features I have already seen implemented at Ad Brouwer. A virtual disk (also called ramdisk) is a part of memory which is laid out as a disk.

The advantage is that you can download all of your systemdisk into this virtual diskmemory after startup. The diskmemory behaves exactly as a real disk, there are directories, tracks, sectors, etc. However, as the information now comes from memory, a virtual disk access has no delay and all utilities are available immediately. Also, your previous systemdrive is now free to do other work, so it will actually save a second drive.

Obviously, the extra memory can also be used as ordinary Ram. The Dos65 team is actually designing a PC-board for the virtual (ram) disk, which will hold 16 Drams of 256K for a total of 512 Mbyte. It will be possible to reduce this to 8 Drams for only 256 Kbyte or to extend the board to 1 Mbyte. Considering the give-away prices for Rams (about Dfl.10) a Ram disk will be a lot cheaper than an extra drive and faster too! As Dos65 supports three drives you can work with the ramdisk and two diskdrives.

According to Erwin Visschedyk the bare pc board will cost about Dfl.50, provided he can find sufficient clubmembers to justify a production order. So don't hesitate to call him

if you want to participate.

By the way, there is an interesting article in the Review Section of the september 1985 Byte issue about a Ramdisk for the Commodore 64 produced by P-Technologies. For Commodore owners worth reading, it provides a good insight in the use of ramdisks.

NEW UTILITIES TOO !

Little by little Dos65 is becoming a very powerful professional tool. Those of my readers who work regularly with the Moser assembler know that this assembler, although rather powerful also tends to be very slow.

To assemble Dos65, for instance, takes at least an quarter of an hour, which is rather bothersome if one has made only a minor adjustment. Ad Brouwer has written a new assembler in C-language for the 6800, which he has now compiled for the 6502.

Although it is just as complete as Moser, its speed is absolutely amazing. Brouwer demonstrated AS65 for me and I estimate the speed at least five times faster than Moser. The source format has been revised and differs considerably from Moser, but everything is very logical.

It is possible to assemble for the 6502, the standard 65C02 and the Rockwell equivalent. In order to convert old Moser source files to the new AS65 format a new utility will be provided called MTOAS (Moser to As).

As65 does not store the object code it produces in memory but writes code directly on disk as binary files. To test the new code it is usually not convenient to load this in its own memory location because there may be eeproms there or you do not wish to overwrite the old code yet. Normally, binary files cannot be loaded to a different starting address than the one provided in the file. Therefore a new utility has been written called OLOAD which allows you to load a binary file elsewhere in memory.

The third complement to AS65 will be a utility called MAP. This will show the begin and end addresses of the various parts of a binary file and also the main start address of this

file. This information is not shown by CAT and I find this utility a handy complement.

Then something we all have been waiting for: a disk doctor. I suppose everybody has sometimes been confronted with the problem of a mutilated disk, the sectors are still there but the directory has been damaged or deleted and the file or the whole disk is no longer accessible. Or just the problem to rename a disk. The Disk Doctor allows you to remedy all this by read and write access to all sectors. Erwin Viisschedyk wrote this utility.

Finally a utility to program the acia on the CPU board will become available.

For the future there are a lot of plans; sometime there will be a C compiler; a legalised Basic and a Pascal are being researched. Some people are working to adapt Elektors graphic card and implement a CAD system! We are at the

beginning of a long development.

Those readers who want more information, would like to contribute or write software, please get in touch with the Dos65 coordinator: Erwin Viisschedyk, Drakesteyn 299, NL-7608 TR Almelo, telephone (31) 5490-71416.

P.S. Time goes by and we keep on forgetting important events. I just read in BYTE's Anniversary Issue: "november 1975, MOS Technology announces the KIM-1 computer, 1K byte Ram, 2K byte monitor in Rom, Keypad, Led readout, cassette and serial output for \$245."

In the same year the 6502 was unveiled and Byte was published for the first time.

It seems a long time ago!

Coen Kleipool.

Siegfried Losensky
Wächtersbach 74
D 6114 Groß-Umstadt

My --SAMSON 65-- is finished and in the meantime I have been busy in working with him to learn BASIC and computing. SAMSON works now very well, but in the beginning I had one not so very fine hardware-problem and it needed a long time of patience and defect-search to solve it. Here is my experience which I want to give to other SAMSON-friends:

Every time I switched line-power on, I had to do it several times to get power-on-reset. Sometimes the first switching was successful, but mostly the third, fourth or even the tenth one. When I finally had the power-on-reset, SAMSON worked without problems.

To find the defect I changed on CPU-Card: C1 three times, C2, IC20=74LS05 into 74LS06, the processor and some other IC's step by step, but all without result. At R17, R18, R19 and D1 I soldered out one side to test them - also without result.

Then a little bit by accident I found the malfunction.

Because I wanted to have my SAMSON more perfect, I decided to change the synchronisation of CPU- and VDU-Card as described at Elektor-Computing No. 2. Therefore I connected IC7(N3)-pin 13 on CPU-Card to pin 30c of the 64-pole-connector and removed the 4-MHz-Quartz. From that time I had no longer the problem described above. The connection of pin 12(IC7) instead of pin 13 brought no positive result.

OCTOPUS

PART 2

```

:
: DISKETTE COPIER VERSION 2.2
:   Version 1.0   : original
:   Version 2.0   : mod 0/Okt.85
:   Version 2.1   : code error
:   Version 2.2   : return->basic
:
: MODIFIED FOR SERIAL SYSTEM
: FOR DOUBLE-SIDED DISK-DRIVES
:
: VALID DRIVE-CONFIGURATION:
:   Drive 1  side  A  and  B
:   Drive 2  side  C  and  D
:
: Attention :
:   Changes have to be made, if
:   sides A  and  B  on differ-
:   ent Disk-Drives !!
:
: =====
: Wolfgang Tietsch
: Eilenburger Weg 11
: 6800 Mannheim 31
: =====
: DECEMBER 1985
: =====

```

```

3410      ;
3420      ;
3430      ;
3440      ;
3450      ;
3460 3DDA 206326  SUB3      JSR HOME
3470 3DDD 205427          JSR LDHEAD
3480 3DE0 AD00C0  LAB04    LDA DRA      ;GET DISK STATUS
3490 3DE3 10FB          BPL LAB04      ;LOOP TILL INDEX END
3500 3DE5 AD00C0  LAB05    LDA DRA      ;GET DISK STATUS
3510 3DE8 30FB          BMI LAB05      ;LOOP TILL INDEX START
3520 3DEA A9FC          LDA #$FC      ;SET PIA TO MODE TO
3530 3DEC 2D02C0          AND DRB      ;WRITE
3540 3DEF 8D02C0          STA DRB      ;DATA TO DISK
3550 3DF2 AD00C0  LAB06    LDA DRA      ;GET DISK STATUS
3560 3DF5 10FB          BPL LAB06      ;WAIT TILL INDEX
3570 3DF7 A201          LDX #$01
3580 3DF9 207A26          JSR DELAY     ;WAIT SOME CYCLES
3590 3DFC A625          LDX ZR05      ;READ 3 BYTES OF TRACK
3600 3DFE 20C227          JSR BYTWRT   ;ZERO-HEADER TO DISK
3610 3E01 A626          LDX ZR06
3620 3E03 20C227          JSR BYTWRT
3630 3E06 A627          LDX ZR07
3640 3E08 20C227          JSR BYTWRT
3650 3E0B A000          LDY #$00
3660 3E0D B1FE  LAB07    LDA (MEMLO),Y
3670 3E0F AA          TAX
3680 3E10 20C227          JSR BYTWRT
3690 3E13 C8          INY
3700 3E14 D0F7          BNE LAB07
3710 3E16 E6FF          INC MEMHI     ;NEXT PAGE
3720 3E18 C627          DEC ZR07      ;DECREMENT PAGE-COUNT
3730 3E1A D0F1          BNE LAB07
3740 3E1C AD00C0  LAB08    LDA DRA      ;GET DISK STATUS

```

3750	3E1F	30FB		BMI LAB08	; WAIT TILL INDEX
3760	3E21	A983		LDA #B3	; RESET WRITE MODE
3770	3E23	206327		JSR UNLOAD	; AND UNLOAD HEAD
3780	3E26	60		RTS	
3790					
3800					
3810					
3820					
3830					
3840					
3850					
3860	3E27	205427	SUB4	JSR LDHEAD	
3870	3E2A	201D27		JSR INDEX	; DO AN INDEX TEST
3880	3E2D	202E27		JSR INACIA	; RESET ACIA
3890	3E30	206C3E		JSR SUB5	; GET FIRST BYTE
3900	3E33	902D		BCC RETA	
3910	3E35	8525		STA ZR05	
3920	3E37	206C3E		JSR SUB5	; GET SECOND BYTE
3930	3E3A	9026		BCC RETA	
3940	3E3C	8526		STA ZR06	
3950	3E3E	206C3E		JSR SUB5	; GET THIRD BYTE
3960	3E41	8527		STA ZR07	; PAGES TO READ
3970	3E43	AA		TAX	; SAVE IT FOR LATER USE
3980	3E44	A000		LDY #00	
3990	3E46	A901	RBYTE	LDA #01	
4000	3E48	2C10C0	STACIA	BIT CACIA	; GET ACIA STATUS
4010	3E4B	F0FB		BEQ STACIA	
4020	3E4D	AD11C0		LDA DACIA	; GET BYTE AND PUT IT
4030	3E50	D1FE		CMP (MEMLO),Y	; COMPARE
4040	3E52	91FE		STA (MEMLO),Y	; INTO MEMORY
4050	3E54	F004		BEQ COMP1	
4060	3E56	A9		.BYTE #A9	
4070	3E57	00	CMPFL2	.BYTE #00	
4080	3E58	F009		BEQ COMP2	
4090	3E5A	C8	COMP1	INY	; NEXT BYTE
4100	3E5B	D0E9		BNE RBYTE	
4110	3E5D	E6FF		INC MEMHI	; NEXT PAGE IN MEMORY
4120	3E5F	CA		DEX	
4130	3E60	D0E4		BNE RBYTE	
4140	3E62	60	RETA	RTS	
4150	3E63	A62B	COMP2	LDX ZR08	; GET STACKPOINTER
4160	3E65	9A		TXS	; UPDATE RETURNADDRESS
4170	3E66	A900		LDA #00	
4180	3E68	48		PHA	
4190	3E69	4C5441		JMP MESERR	; ERROR MESSAGE
4200					
4210					
4220					
4230					
4240	3E6C	18	SUB5	CLC	; GET A BYTE FROM DISK
4250	3E6D	AD00C0	WAIT	LDA DRA	; GET DISK STATUS
4260	3E70	1009		BPL RETB	; RETURN IF INDEX
4270	3E72	AD10C0		LDA CACIA	; GET ACIA STATUS
4280	3E75	4A		LSR A	; RECEIVER BIT IN CARRY
4290	3E76	90F5		BCC WAIT	; LOOP TILL REC. BUF IS FULL
4300	3E78	AD11C0		LDA DACIA	; GET DATA BYTE FROM ACIA
4310	3E7B	60	RETB	RTS	
4320					
4330					

```

4340
4350
4360 3E7C F8      SUB6      SED          ; TRACK UPDATE
4370 3E7D A522     LDA ZR02    ; TRACKNUMB JUST COPIED
4380 3E7F 18       CLC
4390 3E80 6901     ADC #01     ; NEXT TRACK
4400 3E82 D8       CLD
4410 3E83 C523     CMP ZR03    ; LAST TRACK ?
4420 3E85 60       RTS
4430
4440
4450
4460
4470 3E86 38      SUBC      SEC          ; 1. TRACK MINUS 1 DECIMAL
4480 3E87 F8      SED
4490 3E88 E901     SBC #01
4500 3E8A 8522     STA ZR02
4510 3E8C D8       CLD
4520 3E8D 60       RTS
4530
4540
4550
4560
4570
4580
4590
4600
4610
4620 3E8E 20BC26   SUB7      JSR SETTK   ; TRACKNUMBER IS IN ACCU
4630 3E91 201D27   JSR INDEX  ; DO INDEX TEST
4640 3E94 202E27   JSR INACIA  ; RESET ACIA
4650 3E97 A900     LDA #00     ; SET FOR FIRST SECTOR
4660 3E99 852A     STA ZR0A
4670 3E9B E62A     LAB09      INC ZR0A   ; NEXT SECTOR
4680 3E9D 206C3E   SKIP      JSR SUB5   ; GET FIRST BYTE OF TRACK
4690 3EA0 9060     BCC NOBYT   ; BRANCH IF NO BYTE
4700 3EA2 C976     LAB10      CMP #76    ; IS IT A SECTOR HEADER
4710 3EA4 D0F7     BNE SKIP    ; SKIP TRACK HEADER
4720 3EA6 206C3E   JSR SUB5   ; G2. BYTE OF SECTOR HEADER
4730 3EA9 C52A     CMP ZR0A    ; NEXT SECTOR ?
4740 3EAB D0F5     BNE LAB10   ; BRANCH IF NOT
4750 3EAD 206C3E   JSR SUB5   ; GET 3. BYTE (SECTOR LENGTH)
4760 3EB0 AA       TAX        ; PAGES TO READ
4770 3EB1 A429     LDY ZR09
4780 3EB3 C62A     DEC ZR0A    ; SET 0
4790 3EB5 D006     BNE LAB11   ; FIRST SECTOR
4800 3EB7 A522     LDA ZR02    ; ACTUAL TRACK NUMBER
4810 3EB9 99792E   STA BUFFER,Y ; STORE IT
4820 3EBC C8       INY
4830 3EBD 8A       LAB11      TXA        ; NUMBER OF PAGES
4840 3EBE 99792E   STA BUFFER,Y ; STORE IT
4850 3EC1 C8       INY        ; NEXT PAGE
4860 3EC2 8429     STY ZR09    ; OF NEXT SECTOR
4870 3EC4 E62A     INC ZR0A    ; NEXT SECTOR
4880 3EC6 A000     LDY #00
4890 3EC8 A901     LAB14      LDA #01
4900 3ECA 2C10C0   SUACIA     BIT CACIA  ; GET ACIA STATUS
4910 3ECD F0FB     BEQ SUACIA
4920 3ECF AD11C0   LDA DACIA   ; GET BYTE

```

=====

READ DATA OF OTHER TRACKS INTO MEMORY

=====

```

4930 3ED2 7014      BVS PARITY ;CHECK FOR PARITY ERROR
4940 3ED4 D1FE      CMP (MEMLO),Y
4950 3ED6 91FE      STA (MEMLO),Y ;STORE IT
4960 3ED8 F004      BEQ LAB12
4970 3EDA A9        .BYTE $A9      ;LOAD ACCU
4980 3EDB 00        CMPFL1 .BYTE $00      ;COMPARE FLAG
4990 3EDC F01B      BEQ LAB13      ;DO NOT BRANCH IF Z=0
5000 3EDE C8        LAB12 INY          ;NEXT BYTE
5010 3EDF D0E7      BNE LAB14      ;PAGE TRANSFERRED ?
5020 3EE1 E6FF      INC MEMHI     ;NEXT PAGE IN MEMORY
5030 3EE3 CA        DEX           ;NEXT PAGE OF SECTOR
5040 3EE4 D0E2      BNE LAB14      ;ALL PAGES DONE
5050 3EE6 F0B3      BEQ LAB09      ;NO THEN NEXT SECTOR
5060 3EE8 20732D    PARITY JSR STROUT
5070 3EEB 2A        .BYTE '** Parity ** ', $00
5070 3EEC 2A
5070 3EED 20
5070 3EEE 50
5070 3EEF 61
5070 3EF0 72
5070 3EF1 69
5070 3EF2 74
5070 3EF3 79
5070 3EF4 20
5070 3EF5 2A
5070 3EF6 2A
5070 3EF7 20
5070 3EF8 00
5080 3EF9 A62B      LAB13 LDX ZR0B
5090 3EFB 9A        TXS           ;UPDATE RETURN ADDRESS
5100 3EFC A522      LDA ZR02
5110 3EFE 48        PHA
5120 3EFF 4C5441    JMP MESERR
5130 3F02 C62A      NOBYT DEC ZR0A      ;RESET SECTOR COUNT
5140 3F04 F002      BEQ RETC      ;IF ONLY 1 SECTOR THEN RET
5150 3F06 E629      INC ZR09      ;NEXT PAGE
5160 3F08 60        RETC RTS
5170
5180
5190
5200
5210
5220
5230
5240
5250 3F09 205427    SUB8 JSR LDHEAD
5260 3F0C A2FF      LDX #$FF
5270 3F0E 206A2D    LAB15 JSR CRLF
5280 3F11 E8        INX
5290 3F12 BD792E    LDA BUFFER,X ;TRACK NUMBER
5300 3F15 F04A      BEQ LAB17      ;ALL TRACKS COPIED
5310 3F17 862A      STX ZR0A      ;SET SECTOR COUNT
5320 3F19 48        PHA          ;SAVE TRACKNUMBER
5330 3F1A 20732D    JSR STROUT
5340 3F1D 54        .BYTE 'Track ', $00
5340 3F1E 72

```

```

5340 3F1F 61
5340 3F20 63
5340 3F21 6B
5340 3F22 20
5340 3F23 00
5350 3F24 68          PLA          ; GET IT
5360 3F25 48          PHA          ; SAVE IT AGAIN
5370 3F26 20922D      JSR PRTAHX
5380 3F29 68          PLA          ; GET TRACK NUMBER
5390 3F2A 20BC26      JSR SETTK   ; MOVE HEAD TO TR IN ACCU
5400 3F2D A62A        LDX ZR0A    ; SECTORS
5410 3F2F A900        LDA #00     ; BEGINN WITH 1. SECTOR
5420 3F31 8D5E26      STA SECTNM
5430 3F34 E8          INX
5440 3F35 BD792E      LDA BUFFER,X ; GET NUMB OF PAG FOR THIS
5450 3F38 F0D4        BEQ LAB15   ; TRACK
5460 3F3A 862A        STX ZR0A
5470 3F3C 8D5F26      STA PAGCNT  ; NUMB OF PAG FOR TR
5480 3F3F EE5E26      INC SECTNM
5490 3F42 20732D      JSR STROUT
5500 3F45 20          .BYTE ' - ',#00
5500 3F46 2D
5500 3F47 20
5500 3F48 00
5510 3F49 AD5E26      LDA SECTNM
5520 3F4C 20922D      JSR PRTAHX
5530 3F4F 20732D      JSR STROUT
5540 3F52 2F          .BYTE '/ ',#00
5540 3F53 00
5550 3F54 AD5F26      LDA PAGCNT  ; PRINT NUMB OF PAGES
5560 3F57 20922D      JSR PRTAHX
5570 3F5A 20E127      JSR DSKWRT  ; WRITE A SECTOR TO DISK
5580 3F5D A62A        LDX ZR0A    ; SECTORS ALL TRANSFERRED
5590 3F5F D0D3        BNE LAB16   ; NO THEN DO NEXT
5600 3F61 4C6127      JMP UNLDHD  ; UNLOAD HEAD AND RETURN
5610
5620
5630
5640
5650
5660
5670
5680 3F64 A528        SUB9       LDA ZR08
5690 3F66 3005        BMI CHE3    ; DISC FLAG SET - NORMAL
5700 3F68 A521        LDA ZR01
5710 3F6A 4C733F      JMP CHE4
5720 3F6D A521        CHE3       LDA ZR01    ; WHICH DRIVE TO COPY TO
5730 3F6F C520        CMP ZR00    ; IS IT DRIVE TO COPY FROM
5740 3F71 D070        BNE DRIVE
5750 3F73 48          CHE4       PHA          ; SAVE DRIVE NUMBER
5760 3F74 206A2D      JSR CRLF
5770 3F77 20732D      JSR STROUT
5780 3F7A 0A          .BYTE #0A,#0A
5780 3F7B 0A
5790 3F7C 49          .BYTE 'Insert blank',#00
5790 3F7D 6E
5790 3F7E 73
5790 3F7F 65
5790 3F80 72

```

```

5790 3F81 74
5790 3F82 20
5790 3F83 62
5790 3F84 6C
5790 3F85 61
5790 3F86 6E
5790 3F87 6B
5790 3F88 00
5800 3F89 20732D  MESSB      JSR  STROUT
5810 3F8C 20          .BYTE  ' diskette -- press '
5810 3F8D 64
5810 3F8E 69
5810 3F8F 73
5810 3F90 6B
5810 3F91 65
5810 3F92 74
5810 3F93 74
5810 3F94 65
5810 3F95 20
5810 3F96 2D
5810 3F97 2D
5810 3F98 20
5810 3F99 70
5810 3F9A 72
5810 3F9B 65
5810 3F9C 73
5810 3F9D 73
5810 3F9E 20
5820 3F9F 3C          .BYTE  '<RETURN> ?',#00
5820 3FA0 52
5820 3FA1 45
5820 3FA2 54
5820 3FA3 55
5820 3FA4 52
5820 3FA5 4E
5820 3FA6 3E
5820 3FA7 20
5820 3FA8 3F
5820 3FA9 00
5830 3FAA 204023  TRYAG      JSR  INECHO      ;GET ANSWER
5840 3FAD 4B          PHA          ;SAVE IT
5850 3FAE 206A2D      JSR  CRLF
5860 3FB1 6B          PLA
5870 3FB2 C90D      CMP  #00D      ;IS IT A "CR"
5880 3FB4 D0F4      BNE  TRYAG      ;NO THEN TRY AGAIN
5890 3FB6 6B          PLA          ;GET DRIVE NUMBER
5900 3FB7 4CE33F      JMP  DRIVE
5910                ;
5920                ;
5930                ;
5940                ;
5950                ;
5960                ;
5970 3FBA A52B      SUB10      LDA  ZR08
5980 3FBC 3005      BMI  CHE5      ;DISC FLAG SET - NORMAL
5990 3FBE A520      LDA  ZR00
6000 3FC0 4CC93F      JMP  CHE6
6010 3FC3 A520      CHE5      LDA  ZR00      ;IS IT THE SAME DRIVE

```

6020	3FC5	C521		CMP ZR01	;TO WHICH TO COPY
6030	3FC7	D01A		BNE DRIVE	;NO THEN BRANCH
6040	3FC9	48	CHE6	PHA	;SAVE IT
6050	3FCA	206A2D		JSR CRLF	
6060	3FCD	20732D		JSR STROUT	
6070	3FD0	0A		.BYTE #0A,#0A	
6070	3FD1	0A			
6080	3FD2	49		.BYTE 'Insert master',#00	
6080	3FD3	6E			
6080	3FD4	73			
6080	3FD5	65			
6080	3FD6	72			
6080	3FD7	74			
6080	3FD8	20			
6080	3FD9	6D			
6080	3FDA	61			
6080	3FDB	73			
6080	3FDC	74			
6080	3FDD	65			
6080	3FDE	72			
6080	3FDF	00			
6090	3FE0	4C893F		JMP MESSB	
6100	3FE3	20C629	DRIVE	JSR SETDRV	;DRIVE TO COPY FROM
6110	3FE6	206326		JSR HOME	
6120	3FE9	206A2D		JSR CRLF	
6130	3FEC	ADDB41		LDA TEMP2	;SET POINTER TO FIRST
6140	3FEF	85FE		STA MEMLO	;FREE RAM LOCATION
6150	3FF1	ADDA41		LDA TEMP1	
6160	3FF4	85FF		STA MEMHI	
6170	3FF6	60		RTS	
6180					
6190					
6200					
6210					
6220					
6230				=====	
6240				C O P Y MAIN-ROUTINE	
6250				=====	
6260					
6270	3FF7	BA	COPY	TSX	;SAVE STACKPOINTER
6280	3FF8	862B		STX ZR0B	;STORE IT FOR LATER
6290	3FFA	20BA3F		JSR SUB10	;ARRANGE DISC TO COPY FROM
6300	3FFD	A9FF		LDA #*FF	
6310	3FFF	80DB3E		STA CMPFL1	;RESET COMPARE-MODE FLAG
6320	4002	8D573E		STA CMPFL2	
6330	4005	A52D		LDA ZR0D	;1. TRACK TO READ
6340	4007	F006		BEQ ZERTRA	;IF 0 THEN SKIP TRACK 0
6350	4009	20863E		JSR SUBC	;ADJUST 1. TRACK TO COPY
6360	400C	4C1540		JMP ZERTRB	
6370	400F	8522	ZERTRA	STA ZR02	;RESET FIRST TRACK COUNT
6380	4011	20273E		JSR SUB4	;READ TRACK ZERO
6390	4014	08		PHP	;DATA.ON TR 0 ? SAVE IT
6400	4015	207B41	ZERTRB	JSR READ	;READ ALL OTHER TRACKS
6410	4018	20643F		JSR SUB9	;ARRANGE DISC TO COPY TO
6420	401B	A52C		LDA ZR0C	;MAX TRACK NUMBER
6430	401D	F035		BEQ LAB19	;MORE THAN TRACK ZERO
6440	401F	8D7927		STA MAXTRK	;SET FOR TRACKS TO INIT
6450	4022	20732D		JSR STROUT	
6460	4025	49		.BYTE 'Initializing -- ',#00	
6460	4026	6E			

6460	4027	69	
6460	4028	74	
6460	4029	69	
6460	402A	61	
6460	402B	6C	
6460	402C	69	
6460	402D	7A	
6460	402E	69	
6460	402F	6E	
6460	4030	67	
6460	4031	20	
6460	4032	2D	
6460	4033	2D	
6460	4034	20	
6460	4035	00	
6470	4036	A52D	LDA ZR0D ;1. TRACK TO INIT
6480	4038	F012	BEQ INITA
6490	403A	A934	LDA #34 ;MAX TRACK NUMBER
6500	403C	85E5	STA HSTTK
6510	403E	206326	JSR HOME
6520	4041	A52D	LDA ZR0D
6530	4043	20BC26	JSR SETTK ;MOVE HEAD TO TRACK IN ACCU
6540	4046	207227	JSR INIT1 ;INITIALIZE THE TRACKS
6550	4049	4C4F40	JMP INITB
6560	404C	206827	INITA JSR INIT ;INITIALIZE ALL TRACKS
6570	404F	A934	INITB LDA #34 ;SET FOR 35 TRACKS
6580	4051	8D7927	STA MAXTRK
6590	4054	206A2D	LAB19 JSR CRLF
6600	4057	A52D	LDA ZR0D
6610	4059	F003	BEQ ZERTRC
6620	405B	4C6440	JMP LAB20
6630	405E	28	ZERTRC PLP ;WAS THERE ANY DATA ON TR 0
6640	405F	9003	BCC LAB20
6650	4061	20DA3D	JSR SUB3 ;WRITE ON TRACK 0
6660	4064	A529	LAB20 LDA ZR09 ;ANY PAGES READ ALREADY ?
6670	4066	D00D	BNE LAB22 ;BRANCH IF YES
6680	4068	20BA3F	LAB21 JSR SUB10 ;ARRANGE DISC TO COPY FROM
6690	406B	207B41	JSR READ ;READ ALL OTHER TRACKS
6700	406E	A529	LDA ZR09 ;PAGE COUNT
6710	4070	F01B	BEQ LAB23 ;ALL PAGES READ ?
6720	4072	20643F	JSR SUB9 ;ARRANGE DISC TO COPY TO
6730	4075	20732D	LAB22 JSR STROUT
6740	4078	57	.BYTE 'Writing -- ',#00
6740	4079	72	
6740	407A	69	
6740	407B	74	
6740	407C	69	
6740	407D	6E	
6740	407E	67	
6740	407F	20	
6740	4080	2D	
6740	4081	2D	
6740	4082	20	
6740	4083	00	
6750	4084	206A2D	JSR CRLF
6760	4087	20093F	JSR SUB8 ;WRITE TO ALL OTHER TRACKS
6770	408A	4C6840	JMP LAB21
6780			
6790	408D	20732D	LAB23 JSR STROUT

```

6800 4090 0D      .BYTE #0D,#0A
6800 4091 0A
6810 4092 44      .BYTE 'Do you like to compare '
6810 4093 6F
6810 4094 20
6810 4095 79
6810 4096 6F
6810 4097 75
6810 4098 20
6810 4099 6C
6810 409A 69
6810 409B 6B
6810 409C 65
6810 409D 20
6810 409E 74
6810 409F 6F
6810 40A0 20
6810 40A1 63
6810 40A2 6F
6810 40A3 6D
6810 40A4 70
6810 40A5 61
6810 40A6 72
6810 40A7 65
6810 40A8 20
6820 40A9 20      .BYTE ' the discs (Y/N) ? ',#00
6820 40AA 74
6820 40AB 68
6820 40AC 65
6820 40AD 20
6820 40AE 64
6820 40AF 69
6820 40B0 73
6820 40B1 63
6820 40B2 73
6820 40B3 20
6820 40B4 28
6820 40B5 59
6820 40B6 2F
6820 40B7 4E
6820 40B8 29
6820 40B9 20
6820 40BA 3F
6820 40BB 20
6820 40BC 00
6830 40BD 20C53C   JSR GETANS      ;GET ANSWER
6840 40C0 F001     BEQ COMPAR
6850 40C2 60       RTS
6860 40C3 20732D   COMPAR JSR STROUT
6870 40C6 0A       .BYTE #0A,#0A
6870 40C7 0A
6880 40C8 43      .BYTE 'Comparing discettes -- ',#00
6880 40C9 6F
6880 40CA 6D
6880 40CB 70
6880 40CC 61
6880 40CD 72
6880 40CE 69
6880 40CF 6E
6880 40D0 67

```

KENNERS PLEASE!

could'nt you write
in english ? There
are many articles
I would like to
read in the maga-
zine, but it's im-
possible for a
foreigner to read
Dutch.

"Dane"

6880	40D1	20	
6880	40D2	64	
6880	40D3	69	
6880	40D4	73	
6880	40D5	63	
6880	40D6	65	
6880	40D7	74	
6880	40D8	74	
6880	40D9	65	
6880	40DA	73	
6880	40DB	20	
6880	40DC	2D	
6880	40DD	2D	
6880	40DE	20	
6880	40DF	00	
6890	40E0	206A2D	JSR CRLF
6900	40E3	206A2D	JSR CRLF
6910	40E6	A900	LDA #00
6920	40E8	8522	STA ZR02
6930	40EA	A52D	LDA ZR0D
6940	40EC	F006	BEQ ZERTRD
6950	40EE	20863E	JSR SUBC
6960	40F1	4CF740	JMP ZERTRE
6970	40F4	20273E	JSR SUB4 ; READ TRACK 0
6980	40F7	207B41	JSR READ
6990	40FA	A522	LDA ZR02 ; GET LAST TR. NUMBER
7000	40FC	48	PHA ; SAVE IT
7010	40FD	A900	LDA #00 ; SET COMPARING MODE
7020	40FF	8DDB3E	STA CMPFL1
7030	4102	8D573E	STA CMPFL2
7040	4105	8522	STA ZR02
7050	4107	20643F	JSR SUB9 ; ARRANGE DISC TO COPY TO
7060	410A	A52D	LDA ZR0D
7070	410C	F006	BEQ ZERTRF
7080	410E	20863E	JSR SUBC
7090	4111	4C1741	JMP COMP3
7100	4114	20273E	JSR SUB4 ; COMPARE TRACK 0
7110	4117	207B41	JSR READ ; COMPARE OTHER TRACKS
7120	411A	68	PLA ; GET LAST TR NUMBER
7130	411B	48	PHA
7140	411C	C522	CMP ZR02 ; IS IT LAST READ ?
7150	411E	D034	BNE MESERR
7160	4120	C52C	CMP ZR0C ; LAST TR. COPIED
7170	4122	F01D	BEQ MES600
7180	4124	CEDB3E	DEC CMPFL1
7190	4127	CE573E	DEC CMPFL2 ; RESET COMPARE-FLAG
7200	412A	20BA3F	JSR SUB10 ; ARRANGE DISC TO COPY FROM
7210	412D	207B41	JSR READ
7220	4130	A522	LDA ZR02
7230	4132	AA	TAX
7240	4133	68	PLA
7250	4134	8522	STA ZR02 ; UPDATE TRACK FOR COMP.
7260	4136	8A	TXA
7270	4137	48	PHA ; SAVE LAST TR. NUMBER
7280	4138	20643F	JSR SUB9 ; ARRANGE DISC TO COPY TO
7290	413B	EEDB3E	INC CMPFL1 ; SET COMPARE-FLAG
7300	413E	4C1741	JMP COMP3
7310	4141	206A2D	JSR CRLF
7320	4144	20732D	JSR STROUT

7330	4147	47		.BYTE 'Good thru',#00
7330	4148	6F		
7330	4149	6F		
7330	414A	64		
7330	414B	20		
7330	414C	74		
7330	414D	68		
7330	414E	72		
7330	414F	75		
7330	4150	00		
7340	4151	4C6641		JMP MESTRA
7350	4154	20732D	MESERR	JSR STROUT
7360	4157	45		.BYTE 'Error found on',#00
7360	4158	72		
7360	4159	72		
7360	415A	6F		
7360	415B	72		
7360	415C	20		
7360	415D	66		
7360	415E	6F		
7360	415F	75		
7360	4160	6E		
7360	4161	64		
7360	4162	20		
7360	4163	6F		
7360	4164	6E		
7360	4165	00		
7370	4166	20732D	MESTRA	JSR STROUT
7380	4169	20		.BYTE ' track ',#00
7380	416A	74		
7380	416B	72		
7380	416C	61		
7380	416D	63		
7380	416E	6B		
7380	416F	20		
7380	4170	00		
7390	4171	A522		LDA ZR02
7400	4173	20922D		JSR PRTAHX
7410	4176	206A2D		JSR CRLF
7420	4179	68		PLA
7430	417A	60		RTS ; BACK TO MAIN
7440				
7450				
7460				
7470				
7480				
7490				
7500	417B	A900	READ	LDA #\$00
7510	417D	8529		STA ZR09 ; PAGE COUNT
7520	417F	AA		TAX
7530	4180	9D792E	LOOP2	STA BUFFER,X ; FILL BUFFER WITH #00
7540	4183	CA		DEX
7550	4184	D0FA		BNE LOOP2
7560	4186	205427		JSR LDHEAD
7570	4189	207C3E	NEXTRK	JSR SUB6 ; SET FOR NEXT TR. TO COPY
7580	418C	48		PHA ; SAVE NEXT TRACK NUMBER
7590	418D	B046		BCS DONE
7600	418F	ADDB3E		LDA CMPFL1 ; COMPARE-MODE ?
7610	4192	100F		BPL MESCMP
7620	4194	20732D		JSR STROUT

```

7630 4197 52          .BYTE 'Reading ',#00
7630 4198 65
7630 4199 61
7630 419A 64
7630 419B 69
7630 419E 20
7630 419F 00
7640 41A0 4CB141      JMP MESSTR
7650 41A3 20732D      MESCMP      JSR STROUT
7660 41A6 43          .BYTE 'Comparing ',#00
7660 41A7 6F
7660 41A8 6D
7660 41A9 70
7660 41AA 61
7660 41AB 72
7660 41AC 69
7660 41AD 6E
7660 41AE 67
7660 41AF 20
7660 41B0 00
7670 41B1 20732D      MESSTR      JSR STROUT
7680 41B4 54          .BYTE 'Track ',#00
7680 41B5 72
7680 41B6 61
7680 41B7 63
7680 41B8 6B
7680 41B9 20
7680 41BA 00
7690 41BB 68          PLA          ;GET TRACK NUMBER
7700 41BC 48          PHA          ;SAVE IT
7710 41BD 20922D      JSR PRTAHX   ;PRINT IT
7720 41C0 206A2D      JSR CRLF
7730 41C3 68          PLA          ;GET IT AGAIN
7740 41C4 8522        STA ZR02    ;ACTUAL TRACK NUMBER
7750 41C6 208E3E      JSR SUB7    ;READ A TRACK INTO RAM
7760 41C9 A5FF        LDA MEMHI   ;GET LAST PAGE USED
7770 41CB 690C        ADC #0C     ;ADD 12 PAGES FOR SAVETY
7780 41CD CD0023      CMP HIRAM   ;COMPARE TO HIGHEST RAM PAGE
7790 41D0 90B7        BCC NEXTRK  ;READ NEXT TR IF ENOUGH RAM
7800 41D2 4CD941      JMP RETF
7810 41D5 68          DONE      PLA          ;UPDATE STACKPOINTER
7820 41D6 206127      JSR UNLDHD
7830 41D9 60          RETF      RTS
7840 41DA 48          TEMP1     .BYTE $48
7850 41DB 00          TEMP2     .BYTE $00
7860                  .END

```

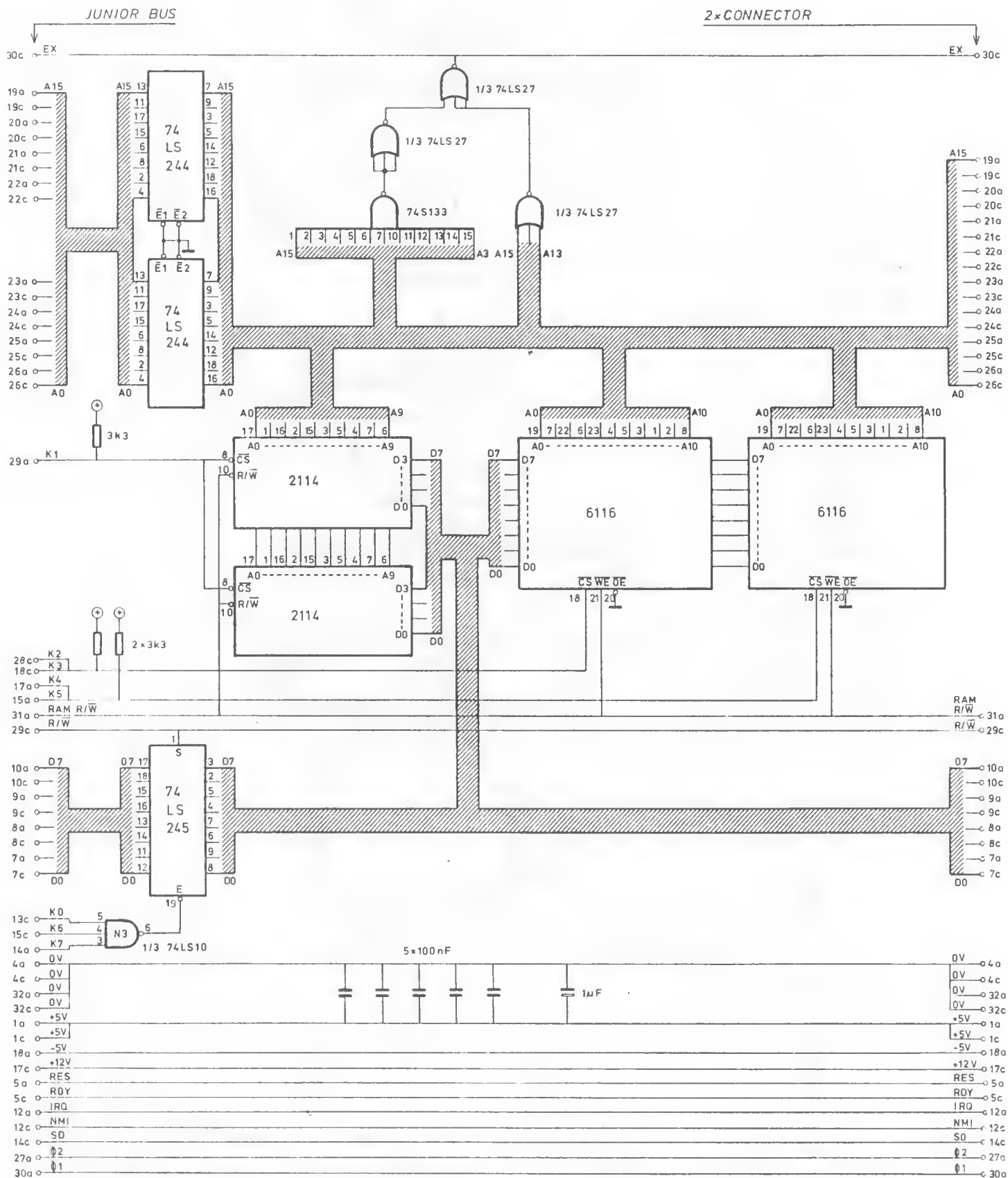
```

*****
*****          ELEKTUUR JUNIOR-COMPUTER HARDWARE          *****
*****          INTERFACEKAART EN GEWIJZIGDE VDU-KAART      *****
*****          Door : Pieter de Visser, Veldhoven          *****
*****
*****
*****

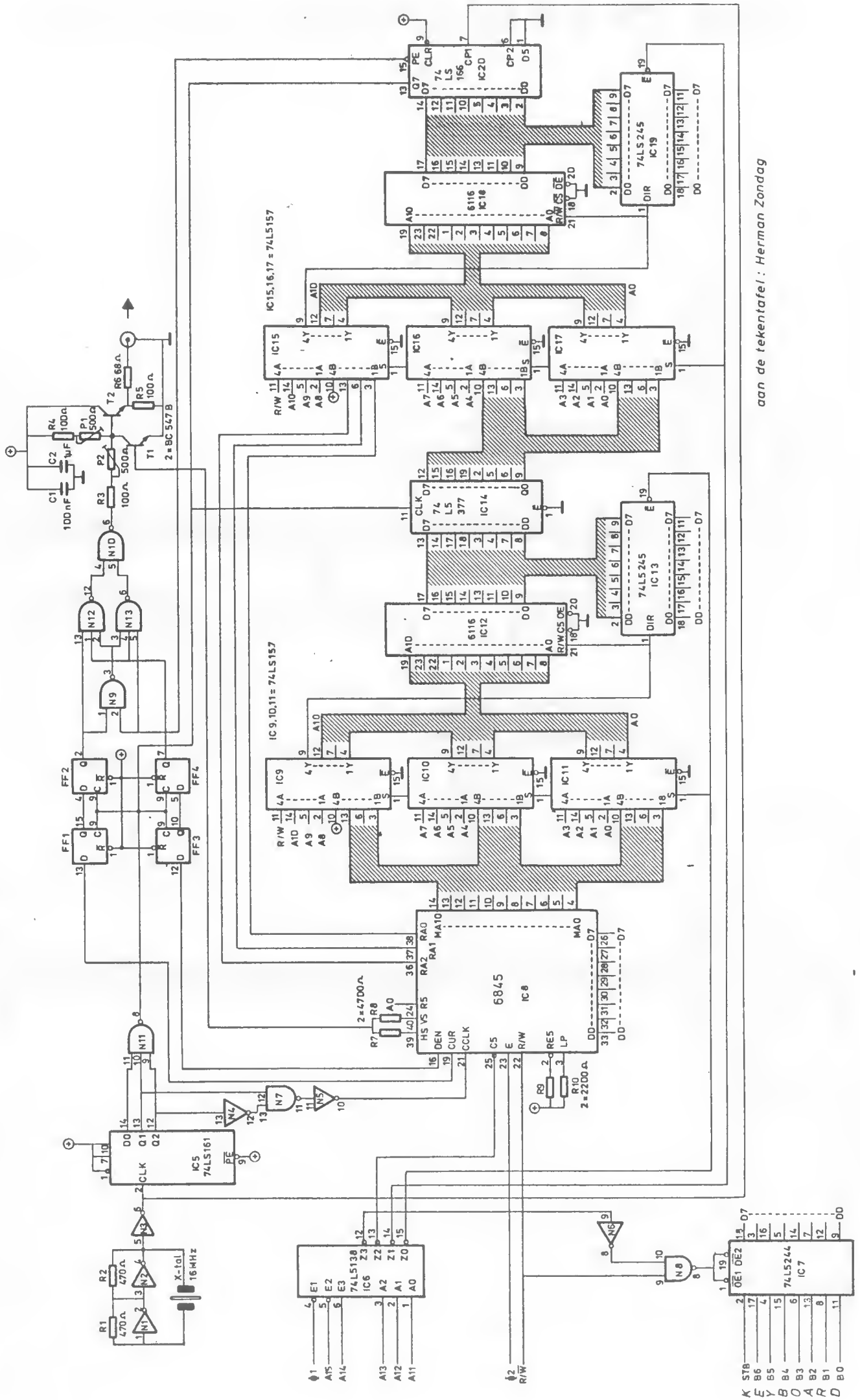
```

Het hart van mijn systeem is de Elektuur Junior-computer. Aan de PIA-connector zit een Apple-compatible cassette-interface met motorsturing. Aan de uitbreidings-connector zit een interface met adres- en data-buffers, en een schakeling voor het opheffen van de achttvoudige adres-sering, en RAM in het geheugenbereik \$400 - \$17FF. Deze interface-kaart biedt wel de

mogelijkheid om op de twee 64-polige connectors een 16K RAM-kaart van Elektuur en nog een, van wat extra's voorziene, VDU-kaart te plaatsen. Deze extra's bestaan uit een in plaats van een 2732 karaktergenerator gebruikte 6116 RAM - zodat ik 256 programmeerbare karakters heb (formaat 8 * 8 pixels) - en een parallelaansluiting voor een ASCII-keyboard.



NB. K0 OP 13c NORMAAL NIET AANWEZIG (EXTRA DRAADBRUG)



aan de tekening: Herman Zondag

***** POSVAL ***** SCHAAKMONITOR *****

uteur : Frans Raaijmakers
Hoogvensestraat 87
5017 CB TILBURG
013 - 366563

stroom : JUNIOR met interface-kaart en 4 K-byte RAM vanaf
adres 2000 of vergelijkbaar systeem.

Is we de populariteit van een gebruikersprogramma mogen afmeten aan het aantal pagina's dat DE 6502 KENNER er aan besteedt, dan staat het schaakprogramma van het Kortekaas met stip op de eerste plaats. Verwonderlijk is dat niet, want iets schept zoveel voldoening als een komputer aan zijn verstand te brengen hoe en intelligent bordspel te spelen. Zo denkt Uw schrijver er in ieder geval over dat anderen daarin mee kunnen gaan, blijkt wel uit de uitbreidingen die inmiddels het daglicht hebben aanschouwd: het programma is geschikt gemaakt voor de JUNIOR, er is een openingsbibliotheek beschikbaar en er is een routine ontwikkeld die de denktijd bekort.

U ligt voor U een positionele evaluatie-routine - Posval - waarmee het schaakprogramma een deel van zijn beslissingen op betere gronden kan nemen. Daarnaast heeft U een schaakmonitor aan; een programma dat oorspronkelijk werd ontwikkeld om het testen van Posval te vergemakkelijken, maar dat ook overigens het bedieningsgemak vergroot. Ik zal mijn verhaal beginnen met een suggestie om de denktijd te bekorten en tevens de analysediepte voor sommige situaties wat te verhogen, maar eerst een vooraf:

het Kortekaas heeft zijn schaakprogramma geschreven onder de voorwaarde dat het niet meer dan 1 K-byte aan geheugen in beslag mocht nemen. In dat licht gezien aan men alleen maar waardering voor zijn programma hebben. Het schrijven van een goed werkende zettengenerator is op zichzelf al geen sinecure en als dat ook nog onder een dergelijke beperkende voorwaarde moet gebeuren, wordt het een bijna onmogelijke opgave. De kanttekeningen die ik hieronder zal maken, moeten dan ook gezien worden als feitelijke konstateringen over de schaaktechnische eigenschappen van het programma. Zij hebben zeker niet de bedoeling de programmatiese restatie te bagatelliseren.

En nog een:

De hier gepresenteerde ontwikkelingen zijn ontwikkeld op een standaard-JUNIOR, uitgebreid met de interfacekaart en een geheugenkaart; een heel eenvoudig systeem vergeleken met de tegenwoordig gangbare hardware. De reden hiervoor is dat de schrijver, ondanks het gemak van beeldschermen en full-screen editors, nog altijd de voorkeur geeft aan een schrijfblok en een potlood. Het programma is geschreven direct in machine-code geschreven.

De vereniging stelt vanzelfsprekend een aantal eisen aan de vorm van een publikatie. Een daarvan is dat het is opge maakt in de vorm die U gewend bent aan te treffen in DE 6502 KENNER. Het was daarom nodig om het programma achteraf om te werken naar een assembly-vorm. Dat is een enorm werk geweest dat gedurende de laatste twee jaren als een joint-venture is uitgevoerd door Uw schrijver en illem van Pelt, aan wie grote dank verschuldigd is omdat zonder zijn enthousiasme en uithoudingsvermogen het programma nooit publikatierijp geworden zou zijn. Om een indruk te geven: als alle proefuitdraaien op elkaar gelegd worden, is de stapel bijna vijf centimeter dik. Alleen al de kosten van de postzegels rukken als een loden last op het budget van de vereniging, dus als de kontributie binnenkort verhoogd wordt, weet U waar het aan ligt.

ENKTIJD EN ANALYSE-DIEPTE

De oorspronkelijke uitvoering heeft het programma ongeveer drie minuten nodig om tot een beslissing te komen. De subroutine Quick versnelt het denkproces aanzienlijk - bij mij met ruim 20% - maar het is mogelijk om nog veel meer tijd te winnen.

Het schaakprogramma werkt met twee limieten - max I en max II - die de zoekdiepte en daarmee de denktijd vastleggen.

Eensimpeld voorgesteld: max II begrenst de zoekdiepte in stellingen waarin een stuk wordt geslagen en max I doet dat in de overige gevallen.

Max I heeft in het schaakprogramma de waarde 2 ply. Nu mag men in beginsel beter schaak verwachten naarmate de zoekdiepte wordt verhoogd, maar de praktijk heeft itgewezen dat een verandering van max I schaaktechnies weinig invloed heeft. In meer dan 90% van de stellingen die ik heb onderzocht komt het schaakprogramma

SPRITES ON ATARI

A sprite is a movable display object. Its shape is different from a character or graphics pixel, due to its independence from other screen activity. A true sprite can pass over any background text or graphics without disturbing the back-ground. It is also usually faster and easier to program than a bit-mapped (high-resolution) shape. Machines with sprites usually include features such as collision-checking (have one or more sprites touched each other?) and variable height and width for the sprites. The Atari 800 has four such sprites, called 'players', and four tiny two-bit sprites called 'missiles' (the missiles can be combined to form a fifth player). They can each be eight bits (dots) wide, and up to 256 lines high. The use of players is not limited to games. They can also form borders, special fall characters, cursors, or even a checkerboard. Other machines that have sprites are the Commodore 64 (with eight 24x21 sprites with multicolor capability and the TI-99/4A).

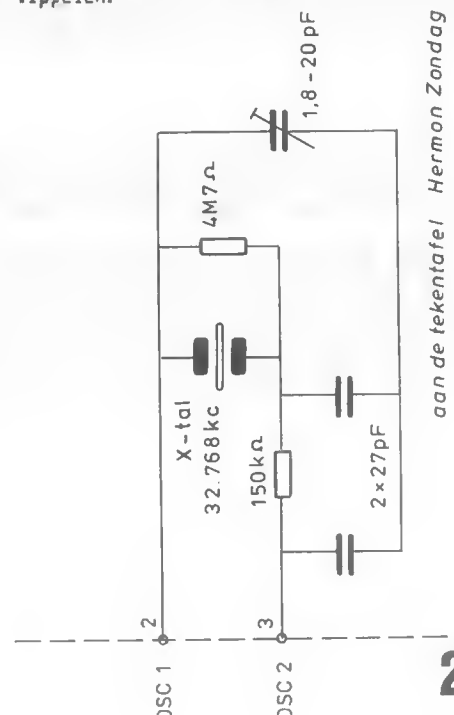
From: Compute! issue 44, vol.6, no.1, Jan. 1984.

PATCH REAL TIME CLOCK ELEKTUUR APRIL 1985

Will Cuijpers, Tholen

De RC 146818 geeft de gebruiker de mogelijkheid uit 3 verschillende klokfrequenties te kiezen: 4.194304 MC, 1.048576 MC en 32.768 KHz. Hierbij geeft de leverancier op dat de opgenomen stroom bij de hoogste frequentie +/- 4 mA en bij de laagste +/- 100 micro-Ampere bedraagt. Om de klok een langere tijd zonder bijladen van de Nicad-akku te kunnen gebruiken spreekt het vanzelf dat dus de laagste klokfrequentie gekozen wordt (32.768 KHz).

Hierbij komt echter een probleem om de hoek kijken. De schakeling als gepubliceerd rondom het kristal werkt prima voor de beide hogere frequenties, doch voor de lage klokfrequentie niet. Na wat uitproberen ben ik op de hieronder volgende schakeling gekomen. Door gebruik te maken van 3 NiCd, allen van 1,2 Volt, elk 110 max. kan ik de klok nu ongemoeid voor maximaal 30 dagen laten tippelen.



ot een en dezelfde beslissing, ongeacht de waarde van max I. Daarbij dient dan ook nog opgemerkt te worden dat in de gevallen waarin een verandering van max I, wel tot een andere zet leidt, er geen enkele garantie is dat die andere zet ook de betere is! Er is wel een verklaring voor dit fenomeen maar die valt buiten het bestek van dit artikel. Waar het hier om gaat is dat een verlaging van max I tot 1 ply weinig invloed heeft op de beslissing terwijl de uitwerking op de denktijd spectaculair is: de gemiddelde denktijd per zet neemt af tot minder dan 30 seconden.

Max II beperkt de zoekdiepte in stellingen waarin een stuk wordt geslagen tot 3 ply, de minimale zoekdiepte waarbij nog enigszins verantwoord schaak te verwachten is. De gevolgen van een zet die verder weg ligt dan de derde ply - achter de horizon, zoals dat heet - worden door het programma niet waargenomen. Het zal duidelijk zijn dat het programma een slagzet beter kan beoordelen naarmate de horizon verder weg ligt, maar ook dat het dan langer duurt om die horizon te bereiken. De tijd die gewonnen wordt door Max I te verlagen kan voor een deel worden geïnvesteerd in een verhoging van Max II, die op 4 en eventueel op 5 ply kan worden gezet. De beste resultaten haalt het programma in mijn teststellingen met Max I = 1 en Max II = 4. De gemiddelde denktijd ligt daarbij net beneden de minuut.

De veranderingen kunnen worden ingevoerd door in de brake-routine op de adressen 2A3C en 2A40 het X-register te laden met de nieuwe waarden. Hier moet wel opgemerkt worden dat schaakzetten niet op dezelfde manier afgehandeld worden als slagzetten. Het programma ziet daardoor af en toe mat in een over het hoofd.

De aanbrengen van de veranderingen verdient het overigens nog steeds aanbeveling om de subroutine Quick te gebruiken. Quick blijft zorgen voor toegevoegde tijdsbesparing van ongeveer 20%.

POSVAL

achtergrond.

Om de bestaansreden van Posval te begrijpen, is het nodig om te weten hoe en op welke gronden het schaakprogramma tot een beslissing komt.

Het schaakprogramma heeft een eenvoudige beslissingsstructuur in de vorm van een trietraps-model waarvan iedere volgende trap pas doorlopen wordt als de vorige niet tot een beslissing heeft geleid. Het programma onderzoekt allereerst de materiele gevolgen van de zet in onderzoek door de waarde van de eigen stukken te vergelijken met die van de andere kleur. Dat ligt ook voor de hand, want in het algemeen heeft de kleur met het beste materiaal ook de beste kansen. De zet die het beste of het minst slechte materiaal oplevert, wordt uitverkoren. Mochten er verschillende 'beste zetten' zijn, dan kiest het programma daaruit de zet die de beste positie oplevert. Zijn er dan nog steeds verschillende mogelijkheden, dan wordt de timer te hulp geroepen om een willekeurige keuze te maken.

Het schaakprogramma vat het begrip 'positie' op als de som van de velden die bestreken worden door de kleur aan zet. Een erg beperkte visie, want in het schaak spel betekent 'positie' echt wel iets meer. De tweede trap van het beslissingsmodel komt daarom als eerste voor uitbreiding en verbetering in aanmerking en dat nu is de bestaansreden van Posval.

Posval onderzoekt en waardeert - afhankelijk van de stelling op het bord - ruim 20 aspecten en manipuleert de variabelen N-ZET en O-ZET (de beste zet tot dan toe en de zet in onderzoek) zodanig dat de beste zet volgens Posval gekozen wordt.

oorsprong.

Posval is afgeleid van de evaluatie-functie van het Amerikaanse schaakprogramma JHESS 4.5; een programma waarvan de eerste versie al in de jaren vijftig geschreven werd en dat lange tijd dood en verderf heeft gezaaid in de Amerikaanse roernooien.

Deze aanpak - het werken vanuit een bestaand programma - bespaart de programmeur letterlijk jaren testwerk. Het grote probleem van iedere evaluatie-functie is immers het vinden van de juiste verhouding tussen het belang van de verschillende elementen die in de beoordeling worden betrokken. Vaak is die juiste verhouding alleen maar te vinden met een trial en error methode; vooral veel error. Een in de praktijk beproefd stelsel, is dan een flinke steun in de rug.

Send your articles and program listing to the Editorial Office, c/o Mr. W.L. van Pelt, Jacob Jordaensstraat 15, NL-2923 CK, Krimpen aan den IJssel, The Netherlands.

BUG

==

Our member Dick Uittenbosch discovered not-printed-lines in the article "GRAPHICS IN APPLESOFT" by Hans Bosch, Twente University of Technology, as published in DE 6502 KENNER Dec. 1985.

The program will run correct by inserting the following lines:

```
131 INTMUL LDA #0
329      LDX #4
395      JSR MOVFM      YA --> FAC
```

Please accept our apology for this omission that could happen by cutting the lines off.

Converting tokenized Basic into regular Basic for C-64

=====

Most computers store Basic statements and commands as tokens. A command such as PRINT (five characters, or bytes, long) is converted into a number (one byte) which the computer can later understand to mean PRINT. This saves memory space. As you enter a Basic program, it is compacted into tokens. Regular Basic is tokenized Basic, but if what you need is a file that contains every letter of every command in ASCII, there is an easy way to do this in Commodore computers. With the program in the computer's memory, type in the following lines:

```
OPEN2,8,2,"program name",S,W":CMD2:LIST
CLOSE2
```

You can insert the name of the file you would like created instead of "program name". After typing the lines and hitting RETURN, you will have a sequential file on your disk that can be read by most wordprocessors. All the letters to commands such as PRINT will be contained in that file.

SUPERTAPE on the C-16

=====

Fred Behringer, Muenchen

One difficulty which I encountered when exploring my newly-bought C-16 (actually, I've bought two because they were so extremely cheap: one for me and one for my son) was the slowness of its tape operation (I understand that this goes with 360 baud ?).

Recently, in "c't", a German computer journal, there was an article by Willi Groebel (c't 1986, issue 2) on the ubiquitous SUPERTAPE program, this time for the C-16, which allows data transfer with up to 7200 baud

al.

SVAL heeft tot doel het schaakprogramma een zeker gevoel voor positie bij te brengen. Het gaat daarbij niet zozeer om rigoureuze schaakconcepten, als wel om eenvoudige vuistregels die iedere schaker toepast zonder er bij na te denken. IJken dienen ontwikkeld te worden; de koning moet verdedigd worden; paarden moeten invloed hebben op het centrum, etc. etc.

Programmastructuur.

SVAL is een volledig zelfstandig werkend programma dat door middel van een speciale patch aan het schaakprogramma wordt geknoopt. Posval maakt geen gebruik van data die door het schaakprogramma worden gegenereerd; deels vanwege hun te beperkte aard en deels omdat die moeilijk bereikbaar zijn. SVAL heeft de gedaante van top-down ontwerp. Ieder soort stuk wordt behandeld door een hoofds subroutine die de naam draagt van dat stuk. De hoofdsroutines zijn zelfstandig werkende programma-onderdelen die zich uitsluitend bedienen van subroutines van een lagere orde. Alleen op het laagste niveau maken sommige subroutines gebruik van elkaars diensten. Het geheel wordt bestuurd door een korte hoofd routine waarin tevens de feitelijke beslissing over de beste zet genomen wordt.

Deze aanpak heeft twee duidelijke voordelen: het programma blijft overzichtelijk, ongeacht hoe lang het wordt en het is mogelijk om iedere hoofds subroutine zonderlijk te testen en eventueel te veranderen zonder de andere programma-onderdelen te beïnvloeden. Daar staat tegenover dat het geheel wat langer wordt en strikt genomen noodzakelijk is.

De hoofdsroutines zijn volgens dezelfde grondgedachte opgebouwd. Na de initialisering van enkele variabelen onderzoekt de subroutine Scan of het stuk op het bord voorkomt. Als het stuk wordt aangetroffen, berekent de hoofds subroutine de positionale waarde die bijgehouden wordt in de stuk teller. Nadat alle aspecten van dat ene stuk aan de orde zijn geweest, keert de hoofds subroutine terug naar het begin om te onderzoeken of het stuk nog vaker voorkomt. Dat gaat zo door totdat Scan meedeelt dat er niet meer stukken van die soort op het bord staan. De stuk teller wordt dan verwerkt in de waardering van de stelling waarna de hoofds subroutine wordt verlaten.

De subroutines van de laagste orde zijn verdeeld in drie groepen van vier. De eerste groep - de reken groep - vervult de rekenkundige functies optellen, vermenigvuldigen en delen. Deze subroutines geven het resultaat uit in een 16-bits formaat. Die zestien bits zijn nodig om voldoende te kunnen differentieren tussen de verschillende positionele aspecten en om te voorkomen dat de tellers overlopen. De tweede groep - de bord groep - vormt een eenvoudige zettengenerator, die voor de snellopende stukken loper, toren en dame de zetmogelijkheden berekent. Deze dubbelure is nodig, omdat het nogal lastig is de veldcontrole van een individueel stuk als afzonderlijk gegeven uit het schaakprogramma te betrekken. Hierdoor kunnen de subroutines van de bord groep nu ook worden ingezet als controles op de functies van de derde groep.

De derde groep - de stuk groep - levert elementaire gegevens die de hoofds routines nodig hebben om efficiënt onderzoek te kunnen doen. De subroutine Afstand berekent de afstand tussen twee willekeurige velden. Eerste veld berekent het eerste id van de lijn waarop het stuk in onderzoek zich bevindt, gezien vanuit de ene naar de andere kant. Status I en Status II tenslotte, leveren informatie over de pionnen. Status I over de pionnen op de lijn waarop het stuk in onderzoek staat en Status II over de aangrenzende lijnen.

Kenmethoden.

SVAL gebruikt in zijn analyses twee manieren van rekenen, die in het programmakommentaar rekenformaat en bitcode worden genoemd.

Het rekenformaat wordt ieder veld voorgesteld door twee cijfercoördinaten van 0 tot 8; B3 = 2,3 en F5 = 6,5 etc. In deze voorstelling kan op een eenvoudige manier worden vastgesteld waar de rand van het bord zich bevindt; die is bereikt als in de coördinaten van het volgende veld een 0 of een 9 voorkomt. Het rekenformaat laat ook toe relatieve betrekkingen tussen twee velden vast te stellen. Het begrip 'afstand' bijvoorbeeld, is gedefinieerd als de absolute som van de verschillen tussen de horizontale en de verticale coördinaten; dat is dus het aantal zetten dat de toren nodig zou hebben als hij maar een veld per beurt zou mogen opschuiven. In het rekenformaat wordt de afstand bepaald door de horizontale en de verticale coördinaten van elkaar af te trekken en de resultaten te sommeren onder verwaarlozing van het teken. Zodoende speelt het teken van de coördinaten geen rol meer. D5 en B3 liggen tenslotte evenver van D4 af. Bitcodes worden vooral gebruikt om de pionninformatie te beoordenen. Een bitcode is een 8-bits woord waarvan de bits de acht velden van de lijn voorstellen.

published in c't for almost any of the existing home computer systems).

One more difficulty I have with my C-16 is its restricted memory capacity which melts down 10 2K if high resolution graphics is used. Up to the present I've seen no chance for a home-brew memory extension since the extension plug is of an exotic measure (can anyone help me?).

So I worked over the supertape program of Willi Groebel in order to keep the memory requirements as low as possible. I succeeded in working out four modified versions:

(1) Supertape, 885 bytes from the top of the memory;

(2) Supertape, residing in the screen color RAM, with the screen window restricted to two (!) lines;

(3) Superload (no SAVE operation), 450 bytes in HiMem;

(4) Superload residing in the screen color RAM, with the screen window restricted to half of its normal size.

If anyone is interested, please do not hesitate to write for a copy via the editorial office.

The organization of these supertape versions is:

SAVE"" 7 = SAVE with 3600 baud
SAVE"" 7,128 = SAVE with 7200 baud
S"" 7,aaaa,bbbb = SAVE machine code from the Monitor
(with 3600 baud only)

The programs worked upon by supertape start with a header. The header is always saved with 3600 baud. Upon LOAD"" 7 the header tells the system whether the program was saved with 3600 or with 7200 baud.

Load machine code programs with L"" 7 using the Monitor. For verification there is VERIFY"" 7 or V"" 7 (for the Monitor).

The Commodore tape operations remain still available if the device number 7, standing for supertape, is omitted.

My experience is that it is almost impossible to save programs with 7200 baud without frequent occurrences of ERROR messages whereas 3600 baud seems to be o.k.

I've also written a utility program by which I can relocate SUPERLOAD to any place in memory (the problem with the C-16 assembler (the Monitor) is that it cannot automatically adapt absolute operands after a relocation).

VERANDERING VAN UW BASICCODE-2 ROUTINE'S

=====

voor Apple-achtigen

Frans Verberkt, Nijmegen

Met regel 210 GET IN\$:RETURN was ik niet zo te vreden, omdat op bepaalde momenten wel eens een toets (per ongeluk) ingedrukt had, voor dat het programma deze routine afvroeg. Soms had dit desastreuze gevolgen. Vandaar dat ik deze regel snel veranderd heb in:

210 POKE 49168,0:GET IN\$:RETURN

Met POKE 49168,0 (\$C010) wordt de input gecleard, d.w.z. dat vanaf dit moment pas een toets verwacht wordt, dus geen misaanslagen meer.

at minst signifikante bit komt overeen met het eerst veld van de lijn, gezien vanuit de kleur aan zet. De positie van het stuk wordt aangegeven door het overnomsomstige bit op 1 te stellen, waarna allerlei aspecten van de formatie kunnen worden onderzocht door de bitcodes te schuiven en er Booleaanse logika mee te schrijven. Een voorbeeld: door de bitcode van de pion in onderzoek eenmaal naar rechts te schuiven en een logiese And-operatie uit te voeren met de bitcodes van de eigen pionnen op de aangrenzende lijnen, komt POSVAL aan de weet of de pion in onderzoek gedekt staat.

perspektieven.

an het schaakprogramma nog verder verbeterd worden? Het antwoord luidt ja en nee. Om met het laatste te beginnen: het programma zal nooit echt goed gaan schaken. Dat zit in de aard van het beest. De zettengenerator is van het brute force type; dat wil zeggen dat tot aan de maximaal toegelaten zoekdiepte alle mogelijke varianten worden opgewekt. Zodoende vindt het programma altijd de beste zet, maar die zekerheid wordt duur betaald: uit tijdoverwegingen is het niet mogelijk om de zoekdiepte groot genoeg te maken om een goede partij schaak op het bord te krijgen.

it Amerikaanse onderzoeken is bekend dat in een gemiddelde schaakstelling meer dan dertig voortzettingen mogelijk zijn. Als we voor het rekengemak even uitgaan van dertig, dan produceert de zettengenerator 30 stellingen op ply 1, 900 op ply 2, 27.000 op ply 3, 810.000 op ply 4 en 24.300.000 op ply 5 etc. De tijd die met het opwekken en verwerken van al die stellingen is gemoeid, neemt vrijwel evenredig met het aantal toe, zodat het wel duidelijk zal zijn dat vanaf de vijfde ply zelfs een computer in de problemen komt als de beslissing binnen drie minuten moet vallen.

och mag pas behoorlijk schaak verwacht worden als het programma in middenspelposities tot vijf a zes ply en in eindspelen minstens tot negen ply mag doorrekenen. Om een zettengenerator binnen een redelijke tijd zulke zoekdieptes te laten bereiken, moet niet alleen de diepte maar ook en met name de breedte van de zettenboom worden beperkt. Helaas is het praktisch niet uitvoerbaar om dit schaakprogramma van een snoeischaar te voorzien. Dat zou zo ingrijpend zijn dat het neerkomt op het schrijven van een nieuw programma en dat kan veel beter gelukken aan de hand van nieuw op te stellen maatstaven.

och zijn er binnen de 'natuurlijke' grenzen van dit programma zeker nog verbeteringen mogelijk.

nlereerst in POSVAL zelf. Een positionele evaluatie-functie is natuurlijk nooit echt klaar. Het grootste gemis in deze versie van POSVAL is het ontbreken van een term 'aangevallen stukken'. Noch het schaakprogramma noch POSVAL genereren een lijst met aangevallen stukken. Daardoor worden stellingen waarin meer dan een stuk tegelijk staat aangevallen, nog weleens verkeerd afgehandeld.

JSVAL zou ook uitgebreid kunnen worden met een term 'slagkracht' om tot uitrukking te brengen dat bijvoorbeeld een loper die een stuk aanvalt sterker is als de dame op dezelfde diagonaal staat.

anzelfsprekend zou een eindspelroutine niet misstaan. De normen die worden aangelegd in het middenspel, zijn vaak niet van toepassing in het eindspel. Het eindspel veronderstelt daarom zijn eigen evaluatieroutine.

erder valt er ook nog te sleutelen aan het schaakprogramma zelf en vooral aan het eerder beschreven beslissingsmodel. Eigenlijk horen de materiaalverhoudingen in de positionele waardering in een term tot uitdrukking te worden gebracht. Nu kijkt het programma het materiaal op de hoogst toegestane zoekdiepte terwijl de positie beoordeeld wordt op de eerste ply. Dat levert soms frikties op.

e manier van materiaalwaardering biedt ook stof tot nadenken. Het schaakprogramma beschouwt de stelling zoals, het die aantreft, als neutraal. Daardoor is het geneigd om tot afruil over te gaan, ook als het al achter staat. Het zou de goede waard zijn om een formule te vinden waarin een bestaand materiaalverschil tot uitdrukking wordt gebracht om te voorkomen dat het programma gaat afruilen als het al achter staat.

et zijn maar een paar suggesties. Het is ongetwijfeld mogelijk om nog veel meer verbeteringen te bedenken maar het effect zal altijd marginaal blijven. De horizontoneffekten die het gevolg zijn van de geringe zoekdiepte verdwijnen er niet voor.

literatuur:

M.B. Immerzeel, Microprocessors van A tot Z, met daarin een uitleg over de werking van de rekenroutines.

Peter W. Frey, Chess Skill in Man and Machine, met onder meer een beschrijving van de evaluatiefunctie van het programma Chess.

H.J. van den Herik, Computerschaak, schaakwereld en Kunstmatige Intelligentie, waarin tal van aspecten van het computerschaak aan de orde komen.

Maar na deze wijziging liep routine 200 niet meer perfect, omdat deze programmaregel juist wel een (vroegere) toetsaanslag verwacht, dus heb ik het hele zaakje veranderd in:

```
205 GET IN$:RETURN
210 POKE 49168,0:GOTO 205
```

Indien U deze wijziging ook wilt doorvoeren, adviseer ik U om bovenstaande regels in te tikken nadat het subroutineblok is klaar gezet (optie 1 van het hoofdmenu).

Save het programma dan op cassette of disk. U heeft van nu af de mogelijkheid en vrijheid, voordat U een Basicode-programma inleest:

- of het originele subroutineblok te gebruiken
- of deze wijziging te laden.

Gebruik beide versies eens afwisselend en maak voor Uzelf uit wat U gebruikersvriendelijk vindt.

6502/6510 VERSCHILLEN

Gebruikers van een Commodore 64 computer die belangstelling hebben voor programmeren in machinetaal, weten misschien niet zeker of hun 6510 verschillen vertoont met de 6502.

De 6502 en de 6510 zijn 'compatible', zoals dat heet. Beide processors gebruiken dezelfde instructieset (LDA, STA etc.) en adresseerformaat (low byte, high byte). Een boek over de 6502 lezen is eigenlijk ook een boek over de 6510 lezen.

De enige belangrijke verschillen tussen de twee processors zijn de bytes 1 en 2 van de 6510. De 6510 maakt het mogelijk om bank-switching toe te passen.

De Commodore 64 heeft 20 K ROM, inclusief de Basic interpreter, Kernal, en Input/Output controle programma's. Er is ook, onder deze ROM, 20 K bruikbare RAM. Je kunt de ROM uitgaan en de RAM in door bank selecting van blokken geheugen. Als je dat zou willen, kun je je C-64 omschakelen in een computer met 64 K bruikbare RAM geheugen door al de ROM uit te schakelen. Maar je zou toch wel je eigen Basic interpreter willen gebruiken, het operating system en de I/O control programma's. Zonder deze zou de computer volledig bevriezen en zou je niet in staat zijn om Basic of machinetaal programma's te schrijven of te runnen.

Een paar problemen

Stefan Sperling, België

Mijn toetsenbord, uitgerust met de AY-3-2376 houdt het achtste bit niet laag, maar brengt het ook hoog bij bepaalde toetsen. Door de draad die naar de computer gaat (bit 8) aan massa te leggen werd het probleem verholpen. Een ander probleem kan de 16 MHz-oscillator zijn op de VDU-kaart die niet wil oscilleren met een kristal, maar wel met de condensatoren en het spoeltje. Dit euvel is te verhelpen door de 74LS04 (IC 3) te vervangen door een 74C04 (dit is nodig om de oscillator te dempen), of IC 3 te vervangen door een 74S04. Bij deze laatste werkte de zaak meteen.

SCHAAKMONITOR

De schaakmonitor is het programma dat de input/output voor zijn rekening neemt. Het schaakprogramma geeft in zijn slotroutine de sturing van de displays en de controle over het toetsenbord over aan de schaakmonitor. Normaal gesproken, lichten alleen de adres-displays op; alleen als in de buffer van de data-displays nuttige informatie staat - een promotiestuk bijvoorbeeld - worden alle displays aangestuurd.

Tijdens het verblijf in de schaakmonitor, houden de numerieke toetsen hun gewone betekenis. De funktietoetsen vervullen de volgende rol :

+ toets	Verplaatst een stuk op het bord.
DA-toets	Plaatst of verwijderd een stuk.
AD-toets	Maakt het bord leeg.
PC-toets	start een nieuwe partij.
GO-toets	wordt genegeerd.

De NMI-toets dient om in het schaakprogramma zelf te komen.

Het is niet mogelijk om via de schaakmonitor ongeldige data in te voeren. Iedere poging daartoe wordt ofwel genegeerd ofwel beantwoord met de melding 'CODE'.

Een stuk wordt verplaatst door het van-veld en het naar-veld in te voeren. De monitor onderzoekt of de opgegeven velden geldig zijn en tevens of er een stuk op het van-veld staat. Een ongeldige invoer geeft de melding 'CODE'.

Let wel : deze funktie verzet stukken buiten de partij om. Zetten die de speler in de partij doet, worden ingegeven via de NMI-toets.

Voor het uitvoeren of terugnemen van een rokade, moet zowel de beweging van de koning als van de toren worden ingegeven.

Met de DA-toets kunnen stukken op het bord worden geplaatst of worden verwijderd. Eerst wordt het veld en daarna de kode van het stuk ingegeven. F4-C5-DA plaatst een zwarte loper op F4 en F4-00-DA maakt F4 leeg.

De monitor controleert ook hier de geldigheid van de opgegeven velden en stukken. Ongeldige velden en stukken leveren weer de melding 'CODE' op.

De AD-toets maakt het bord leeg waarna met behulp van de DA-toets een stelling ingegeven kan worden. Gebruik deze toets alleen als de komputer minstens één zet in de partij heeft gedaan. Anders zijn de data die aangeven welke kleur aan zet is, niet goed gedefinieerd.

De PC-toets zorgt voor het begin van een nieuwe partij. Na PC verschijnt de melding 'FFFF' in de displays waarmee Thor vraagt welke kleur hij moet spelen. Na een 3 speelt Thor wit, na een 0 zwart. In dit laatste geval tonen de displays '0000' waarmee Thor aangeeft dat hij wacht op de eerste zet van wit. Andere toetsen dan 0 of 3 worden genegeerd.

GEBRUIK EN PRAKTIJSE WENKEN

Het hele programma ligt aaneengesloten in het geheugen vanaf hex 2000. Thor maakt gebruik van de zero-page en natuurlijk van de stack. Het is niet nodig om de data van de zero-page in te toetsen. De PC-monitor-routine initieert de zero-page aan het begin van ieder partij.

Het startadres van de monitor ligt op hex 2BAD. Eenvoudshalve is op hex 2000 een sprong naar dit adres opgenomen. Na het inlezen wordt het programma op hex 2000 gestart waarna een PC-kommando zorgt voor het begin van de eerste partij.

Zetten worden op de gebruikelijk manier ingegeven; eerst het vanveld en dan het naarveld in de gewone notatie. Omdat het hexadecimale toetsenbord niet voorziet in de letters 'G' en 'H' worden hiervoor de cijfers '0' en '1' gebruikt.

Rokades worden ingegeven als de beweging van de koning. De korte rokade voor wit is dus E1-O1. De zet wordt ingevoerd met een NMI. Thor controleert de geldigheid van de ingegeven zet. Als de zet geldig is, wordt deze uitgevoerd waarna Thor een antwoord bedenkt. Indien de zet om wat voor reden dan ook ongeldig is, deelt Thor dit mee door de boodschap 'CODE FF' op de displays te zetten. Mat en pat worden eveneens met deze boodschap aangegeven. Tijdens de partij kan de stelling op de eerder beschreven wijze veranderd worden met behulp van de DA-toets en de + toets.

Om een promotie in te geven, is het nodig de schaakmonitor te verlaten door middel van een reset. De kode van het promotiestuk wordt op hex 0 geplaatst waarna de zet op de normale manier kan worden ingegeven. Omdat Thor wordt aangeroepen met een NMI, is het niet nodig terug te keren naar de schaakmonitor voor het ingeven van de pionzet.

Promoties van Thor worden wel via de monitor afgewerkt. De kode van het promotiestuk verschijnt in de rechter displayposities.

In de hier gegeven listing zijn alleen Posval en de schaakmonitor aan het oorspronkelijk programma gekoppeld. de verbinding met Posval staat op hex 2B26. Het verblijf in Posval wordt aangegeven doordat de horizontale segmenten van het meest rechtse display oplichten.

Uiteraard kunnen de uitbreidingen 'Openingen' en 'Quick' die in nummer 20 van de 6502-Kenner zijn gepubliceerd, ook gebruikt worden. De patch voor het openingenprogramma moet worden aangebracht op hex 2A44.

Op Quick te kunnen gebruiken moet men twee patches invoeren :
hex 280A (A9 C0 85 08) wordt JSR Quick - NOP
hex 2876 (C9 41 F0 1F) wordt Jmp Quick II - NOP.

Om Thor te draaien met een ascii-bord en een beeldscherm, zal men zelf de I/O routines moeten schrijven. Veel heeft dat niet om het lijf. de volgende informatie is van belang :

Thor veronderstelt dat de zet van de speler te vinden is in de display-buffers hex FB (van-veld) en hex FA (naarveld). De 'G' wordt voorgesteld door een '0' en de 'H' door een '1'. Een eventueel promotiestuk moet op hex 0 staan. Thor wordt aangeroepen door een NMI of door een jump naar hex 2B7F.

Als Thor zijn tegenzet uitgeeft of als er een foutmelding gedaan wordt, staat de informatie eveneens in de display-buffers hex FB en FA. Een eventueel promotiestuk staat in hex F9. Thor geeft de controle over aan de schaakmonitor door de jump op hex 2A88. Die sprong kan natuurlijk ook naar een andere output-routine wijzen.

DEMONSTRATIE-PARTIJ

Om het verschil tussen het oorspronkelijk schaakprogramma en de versie met Posval te demonstreren, volgt hier een partij die werd gespeeld tussen Thor I en een 'echte' schaakcomputer, de Mephisto II. Omdat de demonstratie bedoeld is de lezer er toe te brengen het programma ook in te toetsen, is uiteraard de mooiste partij uitgekozen. De Mephisto speelt op het laagste nivo. Dat is het enige nivo waarop Thor I al niet in de beginfase overklast wordt. Thor I speelt met de limieten max I=1 en Max 2=3.

De eerste zeven zetten maken beide programma's uit hun openingsboek. Daarna moeten ze op eigen kracht verder.

Na zijn stukken ontwikkeld te hebben, zet Thor op de 24^e zet een aanval in. Ondanks de wat onoverzichtelijke stelling, slagen beide programma's erin zonder materiaalachterstand uit deze fase te komen. Zet 31 van wit is de crux van de partij. Het ontgaat Thor volkomen dat de witte d-pion dreigt te gaan promoveren.

Nadat het zwarte paard weer in positie is gebracht, gaat de strijd verder. In de afwikkeling die begint met zet 37 van zwart, denkt Thor uiteindelijk een pion te kunnen winnen met 43 Tc2 x g2 +. Pas als wit zijn d-pion naar d7 heeft gespeeld, ontdekt Thor dat hij niets meer kan doen aan de promotie. Hij slaagt er dan geruime tijd in om dat probleem over zijn horizon te schuiven - dat wil zeggen verder weg te leggen dan de diepte van zijn analyse - door steeds maar schaak te geven. Dat lukt tot en met zet 49 waarna er weer een witte dame op het bord komt. Terloops wint wit ook nog het zwarte paard dat geen vluchtvelden meer heeft.

Dan rest nog slechts de matvoering zelf. De partij wordt besloten met een komiekeblunder van wit waardoor de winst hem op het laatste nippertje ontgaat.

Wit : Mephisto II

Zwart : Thor I

1	e2 - e4	e7 - e6	
2	d2 - d4	d7 - d5	
3	Pb1 - c3	Pg8 - f6	
4	Lc1 - g5	Lf8 - e7	
5	e4 - e5	Pf6 - d7	
6	Lg5 x e7	Dd8 x e7	
7	f2 - f4	0 - 0	
8	Pg1 - f3	Pb8 - c6	
9	Lf1 - b5	f7 - f5	
10	e5 x f6 (e.p.)	Pd7 x f6	
11	0 - 0	Pc6 - d8	Beducht voor de dubbelpion op de c-lijn.
12	Pf3 - g5	c7 - c6	Voorbarige paardmanoeuvre van wit.

13	Lb5 - d3	h7 - h6	
14	Pg5 - f3	Lc8 - d7	
15	Tf1 - e1	Pd8 - f7	
16	b2 - b3	Pf7 - d6	
17	a2 - a4	Pf6 - g4	Om de toren sterker te maken.
18	Dd1 - d2	a7 - a5	
19	Kg1 - f1 ?	h6 - h5	Onverklaarbare koningsbeweging van wit.
20	Ta1 - d1	De7 - f6	Op jacht naar de f-pion.
21	Pc3 - e2	Df6 - e7	Zwart ziet even geen voortgang.
22	Pe2 - g3	De7 - e8	
23	c2 - c3	De8 - f7	Waarom dan niet meteen.
24	Pg3 - e2	Pd6 - e4	Zwart zet de aanval in.
25	Ld3 x e4	d5 x e4	
26	Pf3 x g5	Pg4 x h2 +	
27	Kf1 - g1	e4 - e3	
28	Dd2 x e3	Ph2 - g4	Wit vermijdt de grote afruil na Pg5 x f7.
29	De3 - e4	Df7 - f5	
30	De4 x f5	e6 x f5	
31	d4 - d5	Ta8 - e8	Ziet het gevaar van de oprukkende d-pion niet
32	d5 - d6	c6 - c5	
33	Kg1 - h1	c5 - c4	Weer een vreemde koningszet van wit.
34	Pe2 - d4	Te8 x e1 +	
35	Td1 x e1	Pg4 - f2 +	
36	Kh1 - g1	Pf2 - d3	
37	Te1 - f1	c4 x b3	Nu begint de grote schoonmaak.
38	Pd4 x b3	Ld7 x a4	
39	Pb3 x a5	Tf8 - c8	
40	c3 - c4	Tc8 - c5	
41	Pa5 x b7	Tc5 x c4	
42	Pg5 - e6	Tc4 - c2	
43	Tf1 - d1	Tc2 x g2 +	Wint een pion.
44	Kg1 x g2	La4 x d1	
45	d6 - d7	Pd3 - e1 +	Promotie kan niet vermeden worden. Zwart schuift
46	Kg2 - h2	Pe1 - f3 +	het probleem over zijn horizon door steeds schaak
47	Kh2 - g3	h5 - h4 +	te geven
48	Kg3 - h3	Pf3 - g1 +	
49	Kh3 x h4	Pg1 - f3 +	
50	Kh4 - g3	Kg8 - f7	Nu is het feest voorbij
51	d7 - d8 (D)	Kf7 x e6	
52	Dd8 x d1	Pf3 - e5	Paard heeft geen velden meer
53	f4 x e5	Ke6 x e5	
54	Pb7 - d6	g7 - g5	
55	Dd1 - d3	f5 - f4 +	
56	Kg3 - f3	Ke5 - f6	
57	Dd3 - d5	Kf6 - e7	
58	Dd5 - e5 +	Ke7 - f8	
59	De5 x g5	pat	Wit is te gulzig. Aangewezen is natuurlijk e5-f6,f8-g8,f6-f7,g8-h8d6-f5,g5-g4,f3xg4, f7-g7 mat.

```

0010: DF47          ORG      *DF47
0020:
0030:
0040: *****
0050: ** PRINTER INITIALIZING FOR EC65 **
0060: ** -SAMSON 65- SYSTEM LOYS V3.1 **
0070: *****
0080: 19-01-1986
0090: LEIF RASMUSSEN
0100: PARKVEJ 1
0110: DK-4534 HØRVE
0120:
0130: DEFINITIONS
0140:
0150: DF47          RECCHA  *      *F71D
0160: DF47          RESET   *      *F32F
0170: DF47          OUTABL  *      *2D73
0180: DF47          STROUT  *      *2343
0190: DF47          PRINT   *      *01
0200: DF47          CTRLA   *
0210:
0220: *DDOB and *DDOC must be changed from
0230: *1D, *F7 to *47, *DF to point to this
0240: routine, the prg, is stored on disk 11
0250: (systemdisk) track 12 sector 5.
0260: -----
0270:
0280: DF47 20 1D F7  ASK      JSR      RECCHA  GET A CHR
0290: DF4A C9 D3     CMPIM    CTRLA   IS IT CR ?
0300: DF4C F0 01     BEQ      START   IF YES, ENTER PRON
0310: DF4E 60        RTS
0320:
0330: DF4F A9 09     START    LDAIM    *09      IS PRINTER ON ?
0340: DF51 CD 22 23   BNE      OUTABL  IF NOT, PRINTERMENU
0350: DF54 D0 06     PROFF    LDAIM    *01      IF YES,
0360: DF56 A9 01     STA      OUTABL   TURN OFF PRINTER,
0370: DF58 8D 22 23   RTS        RETURN TO BASIC
0380: DF5B 60
0390:
0400: DF5C 20 2F F3  PRON     JSR      RESET   CLEAR SCREEN
0410: DF5F 20 73 2D   JSR      STROUT  OUTPUT TEXT
0420: DF62 0D         =        *0D
0430: DF63 0A         =        *0A      1> Pica (default)
0440: DF64 31         =        .1
0450: DF65 3E         =        .>
0460: DF66 20         =        .
0470: DF67 50         =        .P
0480: DF68 69         =        .1
0490: DF69 63         =        .c
0500: DF6A 61         =        .a
0510: DF6B 20         =        .
0520: DF6C 28         =        .(
0530: DF6D 64         =        .d
0540: DF6E 65         =        .e
0550: DF6F 66         =        .f
0560: DF70 61         =        .a
0570: DF71 75         =        .u
0580: DF72 6C         =        .1

```

This is a small printer initialising routine for SAMSON's basic on the system " LOYS V3.1 " disk 11 or 12 from ELEKTOR. The routine will be loaded with bootup. When ctrl-A is entered, a menu appears on the screen:

- ```
1> Pica (default)
2> Elite tab. 10
3> Cond. tab. 25
4> NLQ tab. 6
```

When one of these is entered, the printer is turned on and initialised with the chosen format. The next time the ctrl-A is entered, the printer is turned off, Ctrl-A acts like a flip-flop.

The printer-codes used here are EPSON standard, if your printer use another standard, you must change the output strings. ( Remember, allways end with 00 ).

There is just enough room on track 12 sector 5 ( just behind the print&(x,y)). To implement the routine, adrs \$DDOB, 0C must be changed to \$47, DF to point to this routine.

Store the objectcode on adr  
\$DF47, after having CALLED  
track 12.5 on adr. DD00.

Save it with SA 12,5=DD00/3  
The routine can used in direct mode ( fx. listings ),  
or it can be called from a  
basicprogram: DISK"!GO DF4F

● ○ ♣ ♥ ♦ ♠ ♫ ♪ ♭ ♮ ♯ ♪ ♫ ↑

## SAMSON tips

When using BLOCKGRAPHICS it is necessary to remove the 'ninth' line. To get a rock steady picture, it is not enough to change the CTR-register nine to 7, as stated in EC2. The registers must be recalculated.

Use this :

- ```
10 R = 59661
20 POKE R+4,38
30 POKE R+5,0
40 POKE R+7,30
50 POKE R+9,7
```

JUNIOR'S ASSEMBLER

PAGE 02

```

059001 DF73 74      "      't
060001 DF74 29      "      't
061001 DF75 0D      "      $0D
062001 DF76 0A      "      $0A      2> Elite tab. 10
063001 DF77 32      "      '2
064001 DF78 3E      "      '>
065001 DF79 20      "      '
066001 DF7A 45      "      'E
067001 DF7B 6C      "      '1
068001 DF7C 69      "      'i
069001 DF7D 74      "      't
070001 DF7E 65      "      'e
071001 DF7F 20      "      '
072001 DF80 74      "      't
073001 DF81 61      "      'a
074001 DF82 62      "      'b
075001 DF83 2E      "      '
076001 DF84 20      "      '
077001 DF85 31      "      '1
078001 DF86 30      "      '0
079001 DF87 0D      "      $0D
080001 DF88 0A      "      $0A      3> Cond. tab. 25
081001 DF89 33      "      '3
082001 DF8A 3E      "      '>
083001 DF8B 20      "      '
084001 DF8C 43      "      'C
085001 DF8D 6F      "      'C
086001 DF8E 6E      "      'n
087001 DF8F 64      "      'd
088001 DF90 2E      "      '
089001 DF91 20      "      '
090001 DF92 74      "      't
091001 DF93 61      "      'a
092001 DF94 62      "      'b
093001 DF95 2E      "      '
094001 DF96 20      "      '
095001 DF97 32      "      '2
096001 DF98 35      "      '5
097001 DF99 0D      "      $0D
098001 DF9A 0A      "      $0A      4> NLQ tab. 6
099001 DF9B 34      "      '4
100001 DF9C 3E      "      '>
101001 DF9D 20      "      '
102001 DF9E 4E      "      'N
103001 DF9F 4C      "      'L
104001 DFA0 51      "      'Q
105001 DFA1 20      "      '
106001 DFA2 74      "      't
107001 DFA3 61      "      'a
108001 DFA4 62      "      'b
109001 DFA5 2E      "      '
110001 DFA6 20      "      '
111001 DFA7 36      "      '6
112001 DFA8 0D      "      $0D
113001 DFA9 0A      "      $0A
114001 DFAA 00      "      $00
115001
116001 DFAB A9 09    INIT    LDAIM $09    TURN ON PRINTER

```

.....

```

1170: DFAD BD 22 23          STA   OUTABL
1180:
1190: DFBQ 20 1D F7  ANSWER JSR    RECCHA GET A CHR
1200: DFB3 C9 31          CMPIM  #31   IS IT 1>
1210: DFB5 F0 0F          BEQ    PICA   IF YES "PICA"
1220: DFB7 C9 32          CMPIM  #32   IS IT 2>
1230: DFB9 F0 14          BEQ    ELITE  IF YES "ELITE"
1240: DFB8 C9 33          CMPIM  #33   IS IT 3>
1250: DFB8 F0 1E          BEQ    COND  IF YES "COND"
1260: DFBF C9 34          CMPIM  #34   IS IT 4>
1270: DFC1 F0 28          BEQ    NLQ   IF YES "NLQ"
1280: DFC3 4C F7 DF       JMP    RETURN IF NEITHER RETURN
1290:
1300: DFC6 20 73 2D   PICA JSR    STROUT OUTPUT STRING
1310: DFC9 1B         =      #1B
1320: DFCA 40         =      #40
1330: DFCB 00         =      #00
1340: DFCC 4C F7 DF   JMP    RETURN
1350:
1360: DFCF 20 73 2D   ELITE JSR    STROUT OUTPUT STRING
1370: DFD2 1B         =      #1B
1380: DFD3 40         =      #40
1390: DFD4 1B         =      #1B
1400: DFD5 4D         =      #4D
1410: DFD6 1B         =      #1B
1420: DFD7 6C         =      #6C
1430: DFD8 0B         =      #0B
1440: DFD9 00         =      #00
1450: DFDA 4C F7 DF   JMP    RETURN
1460:
1470: DFDD 20 73 2D   COND JSR    STROUT OUTPUT STRING
1480: DFE0 1B         =      #1B
1490: DFE1 40         =      #40
1500: DFE2 1B         =      #1B
1510: DFE3 0F         =      #0F
1520: DFE4 1B         =      #1B
1530: DFE5 6C         =      #6C
1540: DFE6 19         =      #19
1550: DFE7 00         =      #00
1560: DFE8 4C F7 DF   JMP    RETURN
1570:
1580: DFEB 20 73 2D   NLQ JSR    STROUT OUTPUT STRING
1590: DFEE 1B         =      #1B
1600: DFEF 40         =      #40
1610: DFF0 1B         =      #1B
1620: DFF1 7B         =      #7B
1630: DFF2 31         =      #31
1640: DFF3 1B         =      #1B
1650: DFF4 6C         =      #6C
1660: DFF5 06         =      #06
1670: DFF6 00         =      #00
1680:
1690: DFF7 A9 0D       RETURN LDA1M #0D   CRLF
1700: DFF9 20 43 23   JSR    PRINT
1710: DFFC 20 2F F3   JSR    RESET
1720: DFFF 60         RTS      BACK TO BASIC
1730:

```

KOLORATOR

Are there anyone, who are working on software for a screen-dump from the Kolorator graphics card on Samson to a printer ? I am trying to get a hard copy of the grafics to my Epson printer, and I am very interested to hear from others working with the same problem.

Please write to :

Leif Rasmussen
Parkvej 1
4534 Hørve
Danmark

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω x y z 0 1 2 3 4 5 6 7 8 9 A B C D E F

CONTROL CHARACTERS FOR SAMSON'S CHARACTER EPROM

This is a hexdump for Samson's character generator, which will generate the greek alphabet for the first 32 ASCII characters. This becomes very useful when working with the Wordprocessor, because the printer commands will be visible on the screen (if the changes of the Wordprocessor listed in De 6502 Kenner are made)

```

M
HEXDUMP: 6000,6200
1 2 3 4 5 6 7 8 9 A B C D E F
6000: 00 0E 11 11 10 10 10 00 00 00 00 00 00 00 00 00 00
6010: 00 00 19 25 26 26 17 00 00 00 00 00 00 00 00 00 00
6020: 0E 11 11 1E 11 19 16 10 00 00 00 00 00 00 00 00 00
6030: 3F 20 10 0B 10 20 3F 00 00 00 00 00 00 00 00 00 00
6040: 06 09 04 1E 21 21 1E 00 00 00 00 00 00 00 00 00 00
6050: 0E 11 10 0C 10 11 0E 00 00 00 00 00 00 00 00 00 00
6060: 1F 04 0E 11 0E 04 1F 00 00 00 00 00 00 00 00 00 00
6070: 3F 11 10 10 10 10 3B 00 00 00 00 00 00 00 00 00 00
6080: 0C 12 21 3F 21 12 0C 00 00 00 00 00 00 00 00 00 00
6090: 00 00 0B 04 04 05 02 00 00 00 00 00 00 00 00 00 00
60A0: 00 00 36 49 36 00 00 00 00 00 00 00 00 00 00 00 00
60B0: 00 00 12 0E 0A 09 0B 00 00 00 00 00 00 00 00 00 00
60C0: 00 00 10 0B 0B 14 22 23 00 00 00 00 00 00 00 00 00
60D0: 00 00 22 22 32 2D 20 40 00 00 00 00 00 00 00 00 00
60E0: 00 00 12 09 0A 04 00 00 00 00 00 00 00 00 00 00 00
60F0: 00 00 1C 22 22 14 77 00 00 00 00 00 00 00 00 00 00
6100: 00 01 1E 2A 0A 0A 0A 00 00 00 00 00 00 00 00 00 00
6110: 3F 33 21 1E 21 33 3F 00 00 00 00 00 00 00 00 00 00
6120: 00 00 2E 19 19 1E 10 00 00 00 00 00 00 00 00 00 00
6130: 00 00 1F 24 24 24 1B 00 00 00 00 00 00 00 00 00 00
6140: 00 01 1A 24 04 04 04 00 00 00 00 00 00 00 00 00 00
6150: 00 15 15 0E 04 04 04 00 00 00 00 00 00 00 00 00 00
6160: 00 00 31 0A 0C 12 12 0C 00 00 00 00 00 00 00 00 00
6170: 00 00 22 41 49 49 36 00 00 00 00 00 00 00 00 00 00
6180: 00 00 32 0C 0B 14 23 00 00 00 00 00 00 00 00 00 00
6190: 00 00 11 11 2E 04 0B 10 00 00 00 00 00 00 00 00 00
61A0: 16 09 16 10 0C 02 0C 00 00 00 00 00 00 00 00 00 00
61B0: 1E 21 2D 21 2D 21 1E 00 00 00 00 00 00 00 00 00 00
61C0: 00 24 42 FF 42 24 00 00 00 00 00 00 00 00 00 00 00
61D0: 0B 1C 2A 0B 2A 1C 0B 00 00 00 00 00 00 00 00 00 00
61E0: 0E 11 0C 0A 06 11 0E 00 00 00 00 00 00 00 00 00 00
61F0: 06 09 0B 1C 0B 3C 3A 00 00 00 00 00 00 00 00 00 00
6200: 00

```

4

```

FE
1 REM PRINTROUTINE VOOR TELETYPE 110 BAUD
2 REM TERMINAL HEEFT HANDOMSCHAKELING NAAR DEZE WAARDE
3 REM BY GERARD KEET
5 REM RØDENBURG NØ.3
7 REM 1965BL HEEMSKERK
9 REM TEL. 02510-39763
10 PRINT CHR$(12)
80 PRINT CHR$(12); "ALLE STANDEN PRINTEN"
90 INPUT "STANDEN INLEZEN <J/N>: "; K$
95 IF K$="N" THEN 120
100 DATA "DS", "DAMESSENIØREN", "DJ", "DAMESJUNIØREN"
101 DATA "MA", "MEI SJESADSPIRANTEN", "MP", "MEI SJESPUILL EN"
102 DATA "MW", "MEI SJESWEL PEN", "HS", "HERENSENIØREN"
103 DATA "HJ", "HERENJUNIØREN", "JA", "JØNGENSADSPIRANTEN"
104 DATA "JP", "JØNGENSPUILL EN", "JW", "JØNGENSWEL PEN"
105 DATA "DV", "DAMESVETERANEN", "HV", "HERENVETERANEN"
110 GØSUB 18010
120 DIM PR$(12), PL$(12), FR(12, 4), PL(12, 4)
130 PØKE 26,80
140 A=36:B=12:C=12:D=5:E=10
145 F=D+B*(C+E)
150 AI=24576
155 RT=3
160 GØTØ 19110
1000 REM LEES KLASSE IN PRINTTABEL
1010 IF PEEK(AI+P)=48 THEN 1280
1015 GK=0
1020 IF LI THEN 1125
1025 AR=0
1030 FØR I=1 TØ D:PR$(0)=FR$(0)+CHR$(PEEK(AI-1+P+I)):NEXT I
1040 FØR I=1 TØ 12
1045 IF PEEK(AI+P+5+(I-1)*22)=48 THEN 1089
1050 FØR J=1 TØ 12
1052 PR$(I)=PR$(I)+CHR$(PEEK(AI+P+4+J+(I-1)*22)):NEXT J
1060 FØR J=1 TØ 10:H$=H$+CHR$(PEEK(AI+P+16+J+(I-1)*22)):NEXT J
1070 PR(I,1)=VAL(LEFT$(H$,2)):PR(I,2)=VAL(MID$(H$,3,2))
1080 PR(I,3)=VAL(MID$(H$,5,3)):PR(I,4)=VAL(RIGHT$(H$,3)):H$=""
1082 AR=AR+1:GØTØ 1090
1089 I=12
1090 NEXT I
1095 GØSUB 5010:REM SØRT RECHTS
1100 GØTØ 1270
1125 AL=0
1130 FØR I=1 TØ D:PL$(0)=PL$(0)+CHR$(PEEK(AI-1+P+I)):NEXT I
1140 FØR I=1 TØ 12
1145 IF PEEK(AI+P+5+(I-1)*22)=48 THEN 1189
1150 FØR J=1 TØ 12
1152 PL$(I)=PL$(I)+CHR$(PEEK(AI+P+4+J+(I-1)*22)):NEXT J
1160 FØR J=1 TØ 10:H$=H$+CHR$(PEEK(AI+P+16+J+(I-1)*22)):NEXT J
1170 PL(I,1)=VAL(LEFT$(H$,2)):PL(I,2)=VAL(MID$(H$,3,2))
1180 PL(I,3)=VAL(MID$(H$,5,3)):PL(I,4)=VAL(RIGHT$(H$,3)):H$=""
1182 AL=AL+1:GØTØ 1190
1189 I=12
1190 NEXT I
1200 GØSUB 6010:REM SØRT LINKS
1270 P=P+F:GØTØ 1290
1280 GK=-1
1290 RETURN
1400 REM PRINT LINKS EN RECHTS TBV ALLE STANDEN
1410 IF AL>AR THEN 1430

```

```

1420 MX=AR: GØTØ 1440
1430 MX=AL
1440 IF RT+MX+3<=60 THEN 1500
1450 FØR I=RT TØ 68: PRINT: NEXT: RT=1
1500 S$(0)=PL$(0)
1510 GØSUB 4010
1520 FL$(0)=TEKST$
1530 S$(0)=PR$(0)
1535 IF GK THEN 1545
1540 GØSUB 4010
1542 GØTØ 1550
1545 TEKST$=""
1550 PR$(0)=TEKST$
1560 PRINT TAB(4)PL$(0); TAB(41)PR$(0)
1570 PRINT
1580 FØR I=1 TØ MX
1585 IF FL$(I)="" THEN 1598
1586 IF I>=10 THEN 1588
1587 PRINT " ";
1588 PRINT I; FL$(I);
1589 FØR L=1 TØ 4: L$(L)=STR$(PL(I,L))
1590 L$(L)=RIGHT$(L$(L), LEN(L$(L))-1): NEXT L
1591 PRINT SPC(5-LEN(L$(1))); L$(1); SPC(4-LEN(L$(2)));
1592 PRINT L$(2); SPC(5-LEN(L$(3))); L$(3); "- ";
1593 IF LEN(L$(4))=3 THEN 1595
1594 PRINT SPC(3-LEN(L$(4)));
1595 PRINT L$(4);
1598 IF PR$(I)="" THEN 1608
1600 PRINT TAB(40-LEN(STR$(I)))I; PR$(I);
1601 FØR R=1 TØ 4: R$(R)=STR$(PR(I,R))
1602 R$(R)=RIGHT$(R$(R), LEN(R$(R))-1): NEXT R
1603 PRINT SPC(5-LEN(R$(1))); R$(1); SPC(4-LEN(R$(2)));
1604 PRINT R$(2); SPC(5-LEN(R$(3))); R$(3); "- ";
1605 IF LEN(R$(4))=3 THEN 1607
1606 PRINT SPC(3-LEN(R$(4)));
1607 PRINT R$(4);
1608 PRINT
1800 NEXT I
1810 PRINT: PRINT: RT=RT+MX+3
1820 FØR I=0 TØ 12: PR$(I)=""; PL$(I)=""
1830 FØR J=1 TØ 4: PL(I,J)=0: PR(I,J)=0: NEXT J
1840 NEXT I
1970 GØTØ 1990
1980 GK=-1
1990 RETURN
4000 REM BEPAAL KØFTEKST
4010 CAT$=LEFT$(S$(0),2)
4020 RESTØRE
4030 READ C$, TEKST$: IF C$<>CAT$ THEN 4030
4040 IF MID$(S$(0),3,1)="L" THEN 4060
4050 TEKST$=TEKST$+" KLASSE "+RIGHT$(S$(0),2): GØTØ 4090
4060 TEKST$=TEKST$+" DIVISIE "+RIGHT$(S$(0),2)
4090 RETURN
5000 REM SØRT RECHTS
5010 FØR I=1 TØ AR-1
5020 FØR J=I+1 TØ AR
5030 IF PR(I,2)<PR(J,2) THEN 5140
5040 IF PR(I,2)>PR(J,2) THEN 5180
5050 IF PR(I,1)>PR(J,1) THEN 5140
5060 IF PR(I,1)<PR(J,1) THEN 5180
5070 IF PR(I,3)-PR(I,4)<PR(J,3)-PR(J,4) THEN 5140
5080 IF PR(I,3)-PR(I,4)>PR(J,3)-PR(J,4) THEN 5180

```

```

5090 IF PR(I,4)<>0 AND PR(J,4)<>0 THEN 5120
5100 IF PR(I,4)<>0 THEN 5140
5110 GOTO 5180
5120 IF PR(I,3)/PR(I,4)<PR(J,3)/PR(J,4) THEN 5140
5130 GOTO 5180
5140 HH$=PR$(I):PR$(I)=PR$(J):PR$(J)=HH$
5150 FOR K=1 TO 4
5160 H=PR(I,K):PR(I,K)=PR(J,K):PR(J,K)=H
5170 NEXT K
5180 NEXT J
5190 NEXT I
5200 RETURN
6000 REM SØRT LINKS
6010 FOR I=1 TO AL-1
6020 FOR J=I+1 TO AL
6030 IF PL(I,2)<PL(J,2) THEN 6140
6040 IF PL(I,2)>PL(J,2) THEN 6180
6050 IF PL(I,1)>PL(J,1) THEN 6140
6060 IF PL(I,1)<PL(J,1) THEN 6180
6070 IF PL(I,3)-PL(I,4)<PL(J,3)-PL(J,4) THEN 6140
6080 IF PL(I,3)-PL(I,4)>PL(J,3)-PL(J,4) THEN 6180
6090 IF PL(I,4)<>0 AND PL(J,4)<>0 THEN 6120
6100 IF PL(I,4)<>0 THEN 6140
6110 GOTO 6180
6120 IF PL(I,3)/PL(I,4)<PL(J,3)/PL(J,4) THEN 6140
6130 GOTO 6180
6140 HH$=PL$(I):PL$(I)=PL$(J):PL$(J)=HH$
6150 FOR K=1 TO 4
6160 H=PL(I,K):PL(I,K)=PL(J,K):PL(J,K)=H
6170 NEXT K
6180 NEXT J
6190 NEXT I
6200 RETURN
18010 POKE 6777,1
18020 X=USR("&"0B02",0)
18030 X=USR("&"14BC",0)
18040 RETURN
19100 REM PRINTEN ALLE STANDEN
19110 INPUT "DATUM: ";DA$
19120 PRINT CHR$(18)
19130 PRINT "STANDEN PER ";DA$;" VAN *** Ø D I N - H A N D E A L ***"
19140 PRINT:PRINT
19150 RT=3:P=0:LI=-1
19160 GOSUB 1010:REM LEES KLASSE
19170 IF GK THEN 19900
19180 IF NOT LI THEN 19200
19190 LI=0:GOTO 19160
19200 GOSUB 1410:REM PRINT KLASSE
19210 LI=-1:GOTO 19160
19900 IF LI THEN 19980
19910 GOSUB 1410:REM PRINT KLASSE
19980 PRINT CHR$(20)
19990 END

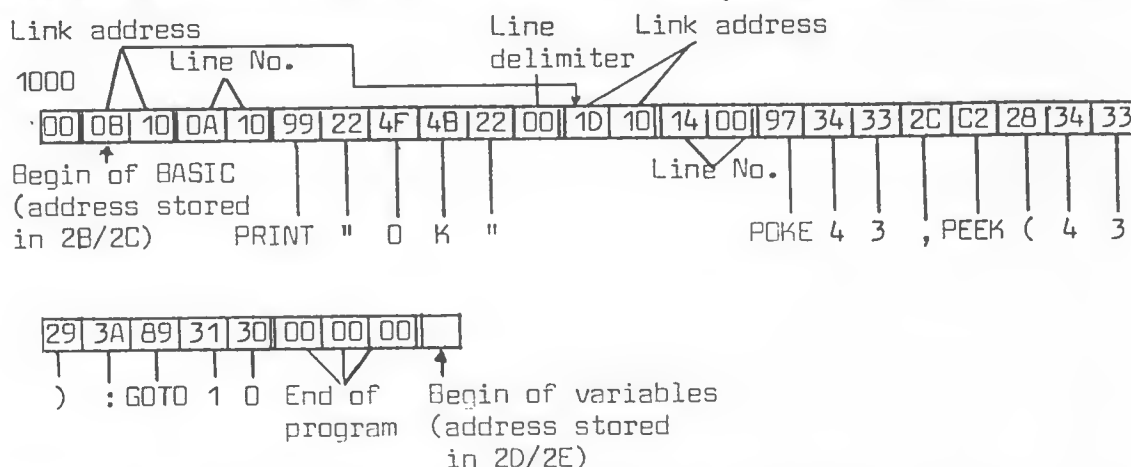
```

ØK

C-16 : Tokenization of the BASIC Instructions

Fred Behringer, Muenchen.

On the Commodore C-16, a BASIC program is stored the same way it is with the VIC-20. Here is an example showing how the BASIC instructions are tokenized:



What follows is a complete list of all C-16 BASIC keyword codes (Red.: see our issue nr. 36, February 1985, p.46/7, Tokenized Microsoft Basic Keywords and addresses C-16). The first list is given with its hexadecimal values only whereas the second list, which is ordered according to increasing code numbers, is given both in hexadecimal and decimal. The correspondences marked with an "!" are the same as with the VIC-20.

ABS	B6	DO	EB	INSTR	D4	PRINT	99	SPC(A6
AND	AF	DRAW	E5	INT	B5	PRINT#	98	SQR	3A
ASC	C6	DSAVE	EF	JOY	CF	PRINTUSING	99FB	SSHAP	E4
ATN	C1	ELSE	D5	KEY	F9	PUDEF	DD	STEP	A9
AUTO	DC	END	80	LEFT\$	C8	RCLR	CD	STOP	90
BACKUP	F6	ERR\$	D3	LEN	C3	RDOT	D0	STR\$	C4
BOX	E1	EXIT	ED	LET	88	READ	87	SYS	9E
CHAR	E0	EXP	BD	LIST	9B	REM	8F	TAB(A3
CHR\$	C7	FN	A5	LOAD	93	RENAME	F5	TAN	C0
CIRCLE	E2	FOR	81	LOCATE	E6	RENUMBER	F8	THEN	A7
CLOSE	A0	FRE	B8	LOG	BC	RESTORE	8C	TO	A4
CLR	9C	GET	A1	LOOP	EC	RESUME	D6	TRAP	D7
CMD	9D	GETKEY	A1F9	MID\$	CA	RETURN	8E	TROFF	D9
COLLECT	F3	GET#	A123	MONITOR	FA	RGR	CC	TRON	D8
COLOR	E7	GO	CB	NEW	A2	RIGHT\$	C9	UNTIL	EC
CONT	9A	GOSUB	8D	NEXT	82	RLUM	CE	USING	EB
COPY	F4	GOTO	89	NOT	A8	RND	BB	USR	B7
COS	BE	GRAPHIC	DE	ON	91	RUN	8A	VAL	C5
DATA	83	GSHAPE	E3	OPEN	9F	SAVE	94	VERIFY	95
DEC	D1	HEADER	F1	OR	B0	SCALE	E9	VOL	DB
DEF	96	HELP	EA	PI	FF	SCNCLR	E8	WAIT	92
DELETE	F7	HEX\$	D2	PAINT	DF	SCRATCH	F2	WHILE	FD
DIM	86	IF	88	PEEK	C2	SGN	B4		
DIRECTORY	EE	INPUT	85	POKE	97	SIN	BF		
DLOAD	F0	INPUT#	84	POS	B9	SOUND	DA		

!80	128	END	!89	137	GOTO	!92	146	WAIT
!81	129	FOR	!8A	138	RUN	!93	147	LOAD
!82	130	NEXT	!8B	139	IF	!94	148	SAVE
!83	131	DATA	!8C	140	RESTORE	!95	149	VERIFY
!84	132	INPUT#	!8D	141	GOSUB	!96	150	DEF
!85	133	INPUT	!8E	142	RETURN	!97	151	POKE
!86	134	DIM	!8F	143	REM	!98	152	PRINT#
!87	135	READ	!90	144	STOP	!99	153	PRINT
!88	136	LET	!91	145	ON	!9A	154	CONT

9B	155	LIST	BD	189	EXP	DF	223	PAINT
9C	156	CLR	BE	190	COS	EO	224	CHAR
9D	157	CMD	BF	191	SIN	E1	225	BOX
9E	158	SYS	CO	192	TAN	E2	226	CIRCLE
9F	159	OPEN	C1	193	ATN	E3	227	GSHAPE
A0	160	CLOSE	C2	194	PEEK	E4	228	SSHAPE
A1	161	GET	C3	195	LEN	E5	229	DRAW
A2	162	NEW	C4	196	STR\$	E6	230	LOCATE
A3	163	TAB(C5	197	VAL	E7	231	COLOR
A4	164	TO	C6	198	ASC	E8	232	SCNCLR
A5	165	FN	C7	199	CHR\$	E9	233	SCALE
A6	166	SPC(C8	200	LEFT\$	EA	234	HELP
A7	167	THEN	C9	201	RIGHT\$	EB	235	DO
A8	168	NOT	CA	202	MID\$	EC	236	LOOP
A9	169	STEP	CB	203	GO	ED	237	EXIT
AA	170	+	CC	204	RGR	EE	238	DIRECTORY
AB	171	-	CD	205	RCLR	EF	239	DSAVE
AC	172	*	CE	206	RLUM	FO	240	DLOAD
AD	173	/	CF	207	JOY	F1	241	HEADER
AE	174	^	DO	208	RDOT	F2	242	SCRATCH
AF	175	AND	D1	209	DEC	F3	243	COLLECT
B0	176	OR	D2	210	HEX\$	F4	244	COPY
B1	177)	D3	211	ERR\$	F5	245	RENAME
B2	178	=	D4	212	INSTR	F6	246	BACKUP
B3	179	<	D5	213	ELSE	F7	247	DELETE
B4	180	SGN	D6	214	RESUME	F8	248	RENUMBER
B5	181	INT	D7	215	TRAP	F9	249	KEY
B6	182	ABS	D8	216	TRON	FA	250	MONITOR
B7	183	USR	D9	217	TROFF	FB	251	USING
B8	184	FRE	DA	218	SOUND	FC	252	UNTIL
B9	185	POS	DB	219	VOL	FD	253	WHILE
BA	186	SQR	DC	220	AUTO	FE	254	----
BB	187	RND	DD	221	PUDEF	FF	255	TT
BC	188	LOG	DE	222	GRAPHIC			

News from the DRAM Front

According to the Süddeutsche Zeitung from 24 March 1986, p.24, Texas Instruments seems to be the first to have succeeded in producing a workable prototype of a 4Mbit DRAM chip. The chip is reported to be organized in 4 blocks of 1 Mbit each. Although SIEMENS have received quite a remarkable amount of subsidy from the German Ministry of Scientific Research and Technology for the purpose of developing such a chip (as far as I have heard) it will still take them years to reach that goal (this is what the Süddeutsche Zeitung reports).

News from Munich: The 1Mbit chip is now on sale for DM 375.- (= Hf1 410.-). It's from Toshiba. In my opinion, it should be forbidden to ask such incredibly high prices - as remarkable as the availability of these chips might appear.

Of course, the result of this development is that I can now get the 41256s (256 kbit) for the equally incredible and low amount of DM 9.- (Hf1 10.-).

The 4164s (64 kbit) are now available for DM 3.50 (Hf1 3.85).

Incidentally, anybody interested in a motherboard full of 4116s (176 chips, 8 rows by 22)? Any offer above postage fee, or things in exchange, will be considered since I didn't spend much money on them. It was the 176 .1µF capacitors on the board I was interested in and which I bought it for.

Fred Behringer
Strassbergerstrasse 9c/519
8000 München 40
Federal Republic of Germany

O T H E L L O

***** SPELREGELS *****

Het spel wordt gespeeld op een 8x8 speelbord. Er wordt gespeeld met schijven die aan de ene zijde wit zijn (*) en aan de andere zijde zwart (O). By het begin van het spel liggen er al 4 schijven op het bord (twee zwarte en twee witte). De ene speler speelt met de witte en de andere met de zwarte schijven. Elk om de beurt leggen de spelers een schijf op het bord. Een schijf dient men zodanig op het bord te leggen dat minstens 1 vijandelijke schijf wordt ingesloten. Dit mas zowel horizontaal, verticaal als diagonaal. Alle schijven die door deze nieuwe schijf ingesloten worden zullen van kleur veranderen. Het doel van het spel is om bij het eind (bord vol, of beide spelers hebben gepast) zoveel mogelijk schijven van eigen kleur te hebben. Veel plezier.



```

10 GOSUB1430:GOTO500
20 T=0
30 FORI=1TO8:FORJ=1TO8
40 IFB(I,J)(<)0THEN60
50 T=1
60 NEXTJ:NEXTI:RETURN
70 GOSUB470
80 PRINT"U SPEELT MET ";J$(K)
90 FORI=1TO8:FORJ=1TO8
100 IFB(I,J)=0THEN140
110 IFB(I,J)=1THEN130
120 PRINT" * ";GOTO150
130 PRINT" O ";GOTO150
140 PRINT" . ";
150 NEXTJ
160 PRINTI;"= X":NEXTI
170 PRINT:PRINT" 1  2  3  4  5  6  7  8 = Y":PRINT:RETURN
180 C1=0:S=0
190 X1=X:Y1=Y:GOSUB400
200 IFT=0THEN390
210 IF B(X,Y)(<)0 THEN 390
220 FORI=-1TO1:FORJ=-1TO1:X1=X:Y1=Y
230 IFI(<)0THEN250
240 IFJ=0THEN380
250 X1=X+I:Y1=Y+J:GOSUB400
260 IFT=0THEN380
270 C2=0
280 IFB(X1,Y1)(<)3-T1THEN320
290 X1=X1+I:Y1=Y1+J:C2=C2+1:GOSUB400
300 IFT=1THEN280
310 GOTO380
320 IFB(X1,Y1)=0THEN380
330 C1=C1+C2
340 IFS=0THEN380
350 X1=X:Y1=Y
360 B(X1,Y1)=T1:X1=X1+I:Y1=Y1+J
370 IFB(X1,Y1)(<)T1THEN360
380 NEXTJ:NEXTI
390 RETURN
400 T=0
410 IFX1(<1)THEN460
420 IFX1(>8)THEN460
430 IFY1(<1)THEN460
440 IFY1(>8)THEN460
450 T=1
460 RETURN
470 GOSUB1430
480 RETURN
490 S1=S1+P3:S1=S1*S1*S1*S1*S1:S1=S1-INT(S1):RETURN
500 DIMB(8,8),E(8,8),J$(2),E1(8,8)
510 GOSUB470
520 INPUT"GEEF EEN WILLEKEURIG GETAL TUSSEN 0 EN 1 (BV 0.5) ";S1
530 RESTORE
540 FORI=1TO8:FORJ=1TO8
550 READE(I,J)
560 GOSUB490
570 E1(I,J)=ABS(E(I,J)+2*S1-1)
580 B(I,J)=0
590 NEXTJ:NEXTI
600 B(4,4)=1:B(4,5)=2:B(5,4)=2:B(5,5)=1
610 J$(1)="O":J$(2)="*"
620 P3=3.141593

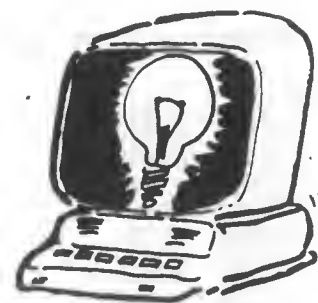
```



```

630 P=0
640 GOSUB470
650 P1=0
660 PRINT"WAT WILT U DOEN ?"
670 PRINT"1. ALS EERSTE SPELEN"
680 PRINT"2. DE MACHINE LATEN BEGINNEN"
690 PRINT"3. DE MACHINE TEGEN ZICHZELF LATEN SPELEN"
700 INPUT"WELKE IS UW KEUZE ";K
710 IFK<1THEN640
720 IFK>3THEN640
730 IFK=2THEN880
740 IFK=3THEN1390
750 GOSUB20
760 IFT=0THEN1150
770 GOSUB70
780 PRINT"WELKE ZET WILT U DOEN (X,Y)?"
790 PRINT"-1,0 WIL ZEGGEN DAT U PAST";
800 INPUTX,Y
810 IFX=-1THEN1290
820 T1=K:GOSUB180
830 IFC1<>0THEN870
840 GOSUB470
850 PRINT"FOUTE ZET. "
860 GOSUB80:GOTO780
870 P=0:S=1:GOSUB190
880 GOSUB70
890 PRINT"IK DENK EVEN NA ";
900 PRINTJ*(3-K);"." :GOSUB20
910 IFT=0THEN1150
920 T1=3-K:V=0
930 FORX=1TO8:FORY=1TO8:GOSUB180
940 IFC1=0THEN1010
950 IFP1=0THEN980
960 IFK=1THEN980
970 S=C1+E1(X,Y)/2:GOTO990
980 S=C1+E(X,Y)/2
990 IFS<VTHEN1010
1000 X2=X:Y2=Y:V=S
1010 NEXTY:NEXTX
1020 IFV=0THEN1090
1030 GOSUB470
1040 P=0
1050 PRINT"IK ZET ";X2;",";Y2;". "
1060 S=1:X=X2:Y=Y2:GOSUB190
1070 IFP1=1THEN1410
1080 GOTO750
1090 GOSUB470
1100 PRINT"IK PAS. "
1110 IFP<0THEN1150
1120 P=1
1130 IFP1=1THEN1410
1140 GOTO770
1150 PRINT"EINDE VAN HET SPEL. "
1160 S=0:V=0
1170 FORI=1TO8:FORJ=1TO8
1180 IFB(I,J)=0THEN1220
1190 IFB(I,J)=KTHEN1210
1200 V=V+1:GOTO1220
1210 S=S+1
1220 NEXTJ:NEXTI:GOSUB70
1230 PRINT"U HEEFT";S;"PUNTEN. IK HEB ER";V;". "
1240 IFS<VTHEN1280
1250 IFS=VTHEN1270
1260 PRINT"HAHA IK WIN !!!":STOP
1270 PRINT"GELIJKSPEL. ":STOP
1280 PRINT"U WINT. ":STOP
1290 IFP=1THEN1150
1300 P=1:GOTO880
1310 DATA10,4,9,8,8,9,4,10
1320 DATA 4,0,2,1,1,2,0,4
1330 DATA 9,2,8,3,3,8,2,9
1340 DATA 8,1,3,0,0,3,1,8
1350 DATA 8,1,3,0,0,3,1,8
1360 DATA 9,2,8,3,3,8,2,9
1370 DATA 4,0,2,1,1,2,0,4
1380 DATA10,4,9,8,8,9,4,10
1390 P1=1:K=2
1400 GOSUB470:GOSUB90:GOTO890
1410 K=3-K:GOTO1400
1420 END
1430 REM SCHOONMAKEN BEELDSCHERM

```



**** V O O R B E E L D ****

U SPEELT MET *

.	1 = X
.	2 = X
.	.	*	0	3 = X
.	.	*	0	0	0	0	.	4 = X
.	.	.	*	*	0	.	.	5 = X
.	*	.	.	6 = X
.	7 = X
.	8 = X

1 2 3 4 5 6 7 8 = Y

WELKE ZET WILT U DOEN (X,Y)
-1,0 WIL ZEGGEN DAT U PAST.

ALS DE * SPELER EEN SCHIJF LEGT
OP DE LIJN 3 VAN DE X-AS EN LYN
6 VAN DE Y-AS DAN ONTSTAAT DE
VOLGENDE SITUATIE.

U SPEELT MET *

.	1 = X
.	2 = X
.	.	*	0	.	*	.	.	3 = X
.	.	*	0	*	*	0	.	4 = X
.	.	.	*	*	*	.	.	5 = X
.	*	.	.	6 = X
.	7 = X
.	8 = X

1 2 3 4 5 6 7 8 = Y

FORMAT LISTER V1.3

FATE 65XX ASSEMBLER V1.0 PAGE: 0001

```

0001 0000 .TIT 'FORMAT LISTER V1.3'
0002 0000 .OPT GEN
0003 0000 .OPT SYM
0004 0000 :
0005 0000 :
0006 0000 : *****
0007 0000 : *
0008 0000 : * MODIFIED FORMAT-LISTER FOR SENIOR MONITOR *
0009 0000 : *
0010 0000 : *****
0011 0000 :
0012 0000 :
0013 0000 : COMMANDS: .TITLE PUTS A TITLE IN HEAD LINE
0014 0000 : .PG FORCES A NEW PAGE START
0015 0000 : === PRINTER TO DOUBLE WIDTH
0016 0000 : *** LIKEWISE. BUT STARTS NEW PAGE FIRST
0017 0000 : +++ PRINTER TO NORMAL WIDTH
0018 0000 : /// LIKEWISE. BUT STARTS NEW PAGE FIRST
0019 0000 : --- PRINTER TO COMPRESSED WIDTH
0020 0000 : \\ LIKEWISE. BUT STARTS NEW PAGE FIRST
0021 0000 : >-( PRINTER TO SHORT LINES
0022 0000 : >+( PRINTER TO LONG LINES
0023 0000 : >6( PRINTER TO 6 LINES/INCH
0024 0000 : >8( PRINTER TO 8 LINES/INCH
0025 0000 : .END ENDS PRINTING OF FILE
0026 0000 :
0027 0000 : ALL PRINTER CONTROL IS MICROLINE 80-A CODE
0028 0000 :
0029 0000 :
0030 0000 : ZERO PAGE DEFINITIONS
0031 0000 :
0032 0000 : *=$0000
0033 0000 :
0034 0000 : COUNTR *==+1 : COUNTER
0035 0001 : TEMPY *==+1 : TEMPORARY FOR Y
0036 0002 : MAXL : MAX. NR. OF LINES IN LISTER
0037 0002 :
0038 0002 : *=$0010
0039 0010 :
0040 0010 : CURCH *==+1 : CURRENT CHARACTER
0041 0011 : CHCNT *==+1 : CHARACTER COUNTER
0042 0012 : NOTSPC *==+1 : NUMBER OF NON-SPACES IN LINE
0043 0013 : CHARS *==+2 : NUMBER OF CHARACTERS IN LINE
0044 0015 : CURPAG *==+2 : CURRENT PAGE NUMBER
0045 0017 : LINE *==+$80 : LINE BUFFER
0046 0097 : CURTIT : CURRENT TITLE
0047 0097 :
0048 0097 : *=$00F0
0049 00F0 :
0050 00F0 : HULP : TEXT POINTER
0051 00F0 :
0052 00F0 :
0053 00F0 : SENIOR MONITOR LOCATIONS
0054 00F0 :
0055 00F0 : *=$0520
0056 0520 :
0057 0520 : INFLG *==+1 : ACTIVE INPUT DEVICE
0058 0521 : OUTFLG *=$057C : ACTIVE OUTPUT DEVICE
0059 057C : SOURCE *==+2 : SOURCE BOTTOM
0060 057E : SORPNT *=$058E : SOURCE POINTER
0061 058E : CURLIN *==+1 : CURRENT LINE NUMBER
0062 058F : MAXLIN : MAXIMUM NUMBER OF LINES
0063 058F :
0064 058F :
0065 058F : EXECUTION VECTOR (START WITH KEY (5))
0066 058F :

```

FORMAT LISTER V1.3

FATE 65XX ASSEMBLER V1.0 PAGE: 0002

```

0067 058F          *=$0506
0068 0506
0069 0506 0002    :      .WOR FORM          : COLD START ENTRY
0070 0508          :
0071 0508          :
0072 0508          :      MAIN PROGRAM
0073 0508          :
0074 0508          *=$0200
0075 0200          :
0076 0200 D8      : FORM      CLD          : JUST IN CASE ...
0077 0201 A2FF          LDX #$FF
0078 0203 9A          : TXS          : RESET STACKPOINTER
0079 0204 A900          LDA #$00
0080 0206 8D8E05      STA CURLIN        : CLEAR CURRENT LINE NUMBER ...
0081 0209 8515          STA CURPAG
0082 020B 8516          STA CURPAG+1      : ... AND CURRENT PAGE NUMBER
0083 020D A93B          LDA #$3B
0084 020F 8502          STA MAXL          : SET MAX. NR. LINES IN LISTER
0085 0211          :
0086 0211          :      INTRODUCTION TO PROGRAM
0087 0211          :
0088 0211 A9B3      : INTRO      LDA #(TEXT1    : PROGRAM INTRODUCTION:
0089 0213 85F0          STA HULP
0090 0215 A903          LDA #)TEXT1        : PROTON FORM-LISTER V1.1
0091 0217 85F1          STA HULP+1
0092 0219 20CC04      JSR MESSO          : PRINT MESSAGE
0093 021C 20F104      JSR CRLF          : CR+LF
0094 021F 20DF04      JSR WHEREI        : DETERMINE INPUT DEVICE
0095 0222 AD2005      LDA INFLG
0096 0225 C94D          CMP #'M'
0097 0227 D003          BNE INTRO1
0098 0229 20FA04      JSR REWIND          : IF MEMORY: RESET SOURCE POINTER
0099 022C 20F104      : INTRO1      JSR CRLF      : CR+LF
0100 022F 20E204      JSR WHEREO        : DETERMINE OUTPUT DEVICE
0101 0232 A90D          LDA #$0D
0102 0234 8597          STA CURTIT      : RESET CURRENT TITLE
0103 0236          :
0104 0236          :      INPUT ONE LINE FROM A.I.D.
0105 0236          :
0106 0236 A900      : FORM1      LDA #$00
0107 0238 8511          STA CHCNT          : CLEAR CHARACTER COUNT
0108 023A C90D      : INL1      CMP #$0D      : CR?, MEANS END OF LINE
0109 023C F00D          BEQ INL2
0110 023E 20E504      JSR INALL          : INPUT CHARACTER FROM A.I.D.
0111 0241 A411          LDY CHCNT
0112 0243 991700      STA LINE.Y          : PUT IT IN LINE-BUFFER
0113 0246 E611          INC CHCNT
0114 0248 4C3A02      JMP INL1
0115 024B C611      : INL2      DEC CHCNT
0116 024D A000          LDY #$00
0117 024F A200      : FORM2      LDX #$00
0118 0251 A903          LDA #$03
0119 0253 20CD03      JSR COMP          : COMPARE COMMAND .PG
0120 0256 D006          BNE NOTPG
0121 0258 201204      JSR HEADER        : YES, FORCE START OF NEW PAGE
0122 025B 4C3602      JMP FORM1
0123 025E A203      : NOTPG      LDX #$03
0124 0260 A906          LDA #$06
0125 0262 20CD03      JSR COMP          : COMPARE COMMAND .TITLE
0126 0265 D03D          BNE NOTTI
0127 0267          :
0128 0267          :      PUT TITLE IN TITLE-BUFFER
0129 0267          :
0130 0267 C8          : INTITL     INY          : GET PAST .TITLE + SPACE
0131 0268 8413          STY CHARS
0132 026A 8C1104      STY TITFL          : SET TITLE FLAG
0133 026D A900          LDA #$00

```

FORMAT LISTER V1.3

FATE 65XX ASSEMBLER V1.0 PAGE: 0003

0134	026F	8512		STA NOTSPC	
0135	0271	A613		LDX CHARS	
0136	0273	B517		LDA LINE.X	
0137	0275	8510		STA CURCH	
0138	0277	A510	INT1	LDA CURCH	
0139	0279	C90D		CMP #\$0D	: CR?. END OF TITLE
0140	027B	F017		BEG INT2	
0141	027D	A412		LDY NOTSPC	
0142	027F	A510		LDA CURCH	
0143	0281	999700		STA CURTIT.Y	: SAVE TITLE CHARACTER
0144	0284	E613		INC CHARS	
0145	0286	E612		INC NOTSPC	
0146	0288	A613		LDX CHARS	
0147	028A	B517		LDA LINE.X	
0148	028C	8510		STA CURCH	
0149	028E	A92D		LDA #\$2D	: TITLE NOT LONGER THEN 45
0150	0290	C513		CMP CHARS	
0151	0292	B0E3		BCS INT1	
0152	0294	A412	INT2	LDY NOTSPC	
0153	0296	A90D		LDA #\$0D	
0154	0298	999700		STA CURTIT.Y	: CLOSE TITLE
0155	029B	EE8E05		INC CURLIN	: INCREMENT CURRENT LINE
0156	029E	202104		JSR HEAD1	: GO PRINT HEAD
0157	02A1	4C3602		JMP FORM1	
0158	02A4	A209	NOTTI	LDX #\$09	
0159	02A6	A903		LDA #\$03	
0160	02A8	20CD03		JSR COMP	: COMPARE COMMAND ***
0161	02AB	D006		BNE NOTST	
0162	02AD	201204		JSR HEADER	: YES, START NEW PAGE ...
0163	02B0	4CDF02		JMP EQUAL	: ... AND SET PRINTER TO WIDE
0164	02B3	A20C	NOTST	LDX #\$0C	
0165	02B5	A904		LDA #\$04	
0166	02B7	20CD03		JSR COMP	: COMPARE COMMAND .END
0167	02BA	D01A		BNE NOTEN	
0168	02BC	20BA04		JSR PRI6L	: SET PRINTER TO 6 LINES/INCH
0169	02BF	20B304		JSR PRLONG	: SET PRINTER TO LONG LINES
0170	02C2	20AB04		JSR PRNORM	: NORMAL CHARACTERS
0171	02C5	A900		LDA #\$00	
0172	02C7	8D1104		STA TITFL	: CLEAR TITLE FLAG
0173	02CA	20EE04		JSR CLOSED	: CLOSE OUTPUT DEVICE
0174	02CD	20EB04		JSR CLOSEI	: CLOSE INPUT DEVICE
0175	02D0	20F104		JSR CRLF	: CR+LF
0176	02D3	4CDC04		JMP COMIN	: BACK TO SENIOR
0177	02D6	A210	NOTEN	LDX #\$10	
0178	02D8	A903		LDA #\$03	
0179	02DA	20CD03		JSR COMP	: COMPARE COMMAND ===
0180	02DD	D006		BNE NOTEQ	
0181	02DF	209804	EQUAL	JSR PRWIDE	: YES, SET PRINTER TO WIDE
0182	02E2	4C4F02		JMP FORM2	: GO DEAL WITH REST OF LINE
0183	02E5	A213	NOTEQ	LDX #\$13	
0184	02E7	A903		LDA #\$03	
0185	02E9	20CD03		JSR COMP	: COMPARE COMMAND ///
0186	02EC	D009		BNE NOTSL	
0187	02EE	201204		JSR HEADER	: YES, START NEW PAGE ...
0188	02F1	20AB04	SLASH	JSR PRNORM	: ... AND SET PRINTER TO NORMAL
0189	02F4	4C4F02		JMP FORM2	: GO DEAL WITH REST OF LINE
0190	02F7	A216	NOTSL	LDX #\$16	
0191	02F9	A903		LDA #\$03	
0192	02FB	20CD03		JSR COMP	: COMPARE COMMAND +++
0193	02FE	F0F1		BEG SLASH	: GO SET PRINTER TO NORMAL
0194	0300	A219	NOTPL	LDX #\$19	
0195	0302	A903		LDA #\$03	
0196	0304	20CD03		JSR COMP	: COMPARE COMMAND ---
0197	0307	D008		BNE NOTMIN	
0198	0309	A91D	MIN	LDA #\$1D	
0199	030B	209A04		JSR MODE	: SET PRINTER TO COMPRESSED MODE
0200	030E	4C4F02		JMP FORM2	: GO DEAL WITH REST OF LINE

FORMAT LISTER V1.3

FATE 65XX ASSEMBLER V1.0 PAGE: 0004

0201	0311	A21C	NOTMIN	LDX ##1C	
0202	0313	A903		LDA ##03	
0203	0315	20CD03		JSR COMP	: COMPARE COMMAND \\\
0204	0318	D006		BNE NOTBSL	
0205	031A	201204		JSR HEADER	: GO TO NEW PAGE
0206	031D	4C0903		JMP MIN	: SET PRINTER TO COMPRESSED MODE
0207	0320	A21F	NOTBSL	LDX ##1F	
0208	0322	A903		LDA ##03	
0209	0324	20CD03		JSR COMP	: COMPARE COMMAND >-(<
0210	0327	D00B		BNE NOTSH	
0211	0329	20AF04		JSR PRESC	
0212	032C	A942		LDA #'B'	: SET PRINTER TO SHORT LINE
0213	032E	209D04	BSL	JSR OUTPO	
0214	0331	4C4F02		JMP FORM2	: GO DEAL WITH REST OF LINE
0215	0334	A222	NOTSH	LDX ##22	
0216	0336	A903		LDA ##03	
0217	0338	20CD03		JSR COMP	: COMPARE COMMAND >+(<
0218	033B	D006		BNE NOTLO	
0219	033D	20B304		JSR PRLONG	
0220	0340	4C4F02		JMP FORM2	: GO DEAL WITH REST OF LINE
0221	0343	A225	NOTLO	LDX ##25	
0222	0345	A903		LDA ##03	
0223	0347	20CD03		JSR COMP	: COMPARE COMMAND >6(<
0224	034A	D006		BNE NOTL6	
0225	034C	20BA04		JSR PRI6L	: SET PRINTER TO 6 LINES/INCH
0226	034F	4C4F02		JMP FORM2	: GO DEAL WITH REST OF LINE
0227	0352	A228	NOTL6	LDX ##28	
0228	0354	A903		LDA ##03	
0229	0356	20CD03		JSR COMP	: COMPARE COMMAND >8(<
0230	0359	D014		BNE NOTL8	
0231	035B	20AF04		JSR PRESC	
0232	035E	A938		LDA #'8'	: SET PRINTER TO 8 LINES/INCH
0233	0360	209D04		JSR OUTPO	
0234	0363	A94E		LDA ##4E	
0235	0365	8502		STA MAXL	
0236	0367	A954		LDA ##54	
0237	0369	8D8F05		STA MAXLIN	
0238	036C	4C4F02		JMP FORM2	: GO DEAL WITH REST OF LINE
0239	036F	A502	NOTL8	LDA MAXL	
0240	0371	CD8E05		CMP CURLIN	: AT MAXIMUM LINE NUMBER?
0241	0374	B009		BCS NOTL8A	
0242	0376	A900		LDA ##00	
0243	0378	C512		CMP NOTSPC	
0244	037A	D003		BNE NOTL8A	
0245	037C	201204		JSR HEADER	: FORCE NEW PAGE
0246	037F	A502	NOTL8A	LDA MAXL	
0247	0381	CD8E05		CMP CURLIN	
0248	0384	B003		BCS NOTL8B	
0249	0386	201204		JSR HEADER	: FORCE NEW PAGE
0250	0389		:		
0251	0389		:	PRINT ONE LINE ON A.O.D.	
0252	0389		:		
0253	0389	98	NOTL8B	TYA	
0254	038A	8513	PRL0	STA CHARS	
0255	038C	C511	PRL1	CMP CHCNT	
0256	038E	F005		BEQ PRL2	
0257	0390	9003		BCC PRL2	
0258	0392	4CA003		JMP PRL3	
0259	0395	A613	PRL2	LDX CHARS	
0260	0397	B517		LDA LINE,X	
0261	0399	C90D		CMP ##0D	: CR?, MEANS END OF LINE
0262	039B	F00A		BEQ PRL4	
0263	039D	20E804		JSR OUTALL	: GO PRINT CHARACTER
0264	03A0	E613	PRL3	INC CHARS	
0265	03A2	A513		LDA CHARS	
0266	03A4	4C8C03		JMP PRL1	
0267	03A7	EE8E05	PRL4	INC CURLIN	

FORMAT LISTER V1.3

FATE 65XX ASSEMBLER V1.0 PAGE: 0005

```

0268 03AA 20F404      JSR RCHEK      : CHECK FOR KEYBOARD INTERRUPT
0269 03AD 20F104      JSR CRLF      : CR+LF
0270 03B0 4C3602      JMP FORM1     : GO DEAL WITH NEXT LINE
0271 03B3             :
0272 03B3             :
0273 03B3             : SUBROUTINES
0274 03B3             :
0275 03B3             : PROGRAM TEXTS
0276 03B3             :
0277 03B3 464F TEXT1   .BYT 'FORMAT-'
0277 03B5 524D
0277 03B7 41542D
0278 03BA 4C49         .BYT 'LISTER V1.3:'
0278 03BC 5354
0278 03BE 4552
0278 03C0 2056
0278 03C2 312E
0278 03C4 333B
0279 03C6             :
0280 03C6 5041 TEXT2   .BYT 'PAGE: :'
0280 03C8 4745
0280 03CA 3A203B
0281 03CD             :
0282 03CD             : COMPARE COMMAND ROUTINE
0283 03CD             :
0284 03CD 8500 COMP    STA COUNTR     : SET COMMAND LENGTH
0285 03CF 8401         STY TEMPY      : SAVE Y FOR NOT EQUAL
0286 03D1 BDE503 COMP1 LDA COMMT,X
0287 03D4 D91700      CMP LINE,Y     : IS LINE STARTING WITH COMMAND?
0288 03D7 D007        BNE COMP2      : NOT THIS COMMAND
0289 03D9 E8          INX
0290 03DA C8          INY
0291 03DB C600        DEC COUNTR     : FULL COMMAND?
0292 03DD D0F2        BNE COMP1
0293 03DF 60          RTS            : Z=1 IF COMMAND FOUND
0294 03E0 08 COMP2   PHP
0295 03E1 A401        LDY TEMPY
0296 03E3 28          PLP
0297 03E4 60          RTS
0298 03E5             :
0299 03E5             : COMMAND TABLE
0300 03E5             :
0301 03E5 2E5047 COMMT .BYT '.PG'
0302 03E8 2E54        .BYT '.TITLE'
0302 03EA 4954
0302 03EC 4C45
0303 03EE 2A2A2A      .BYT '***'
0304 03F1 2E45        .BYT '.END'
0304 03F3 4E44
0305 03F5 3D3D3D      .BYT '=== '
0306 03F8 2F2F2F      .BYT '/// '
0307 03FB 2B2B2B      .BYT '+++'
0308 03FE 2D2D2D      .BYT '--- '
0309 0401 5C5C5C      .BYT '\\\\ '
0310 0404 3E2D3C      .BYT '}<'
0311 0407 3E2B3C      .BYT '}>'
0312 040A 3E363C      .BYT '}>6<'
0313 040D 3E383C      .BYT '}>8<'
0314 0410             :
0315 0410 1E PRMODE   .BYT $1E      : PRINTER MODE: DEF. NORMAL
0316 0411 00 TITFL    .BYT $00      : TITLE FLAG: DEF. NO TITLE
0317 0412             :
0318 0412             : PRINT HEADER IN LISTING
0319 0412             :
0320 0412 8401 HEADER STY TEMPY      : SAVE Y
0321 0414 AD8E05      LDA CURLIN     : IF CURRENT LINE = 0 ...
0322 0417 F008        BEQ HEAD1      : ... NO LINE SKIPPING

```

FORMAT LISTER V1.3

FATE 65XX ASSEMBLER V1.0 PAGE: 0006

```

0323 0419 20FD04      JSR FORMS          : FORCE NEW PAGE
0324 041C AD1104      LDA TITFL           : TITLE FLAG SET?
0325 041F F013        BEQ HEAD2          : NO TITLE
0326 0421 A902        LDA #$02
0327 0423 8D8E05      STA CURLIN         : CURRENT LINE = 2
0328 0426 A91E        LDA #$1E
0329 0428 209D04      JSR OUTPO          : PRINTER TO NORMAL
0330 042B 203704      JSR PRHEAD         : PRINT HEAD
0331 042E AD1004      LDA PRMODE
0332 0431 209D04      JSR OUTPO          : PRINTER AS IT WAS
0333 0434 A401        LDY TEMPY          : RESTORE Y
0334 0436 60          RTS
0335 0437 A900        LDA #$00
0336 0439 8513        STA CHARS          : CLEAR NUMBER OF CHARACTERS
0337 043B C92D        PRH0  CMP #$2D     : MAXIMUM TITLE LENGTH?
0338 043D F005        BEQ PRH1
0339 043F 9003        BCC PRH1
0340 0441 4C5604      JMP PRH3
0341 0444 A613        PRH1  LDX CHARS
0342 0446 B597        LDA CURTIT,X       : GET TITLE CHARACTER
0343 0448 C90D        CMP #$0D           : WAS IT CR?
0344 044A F00A        BEQ PRH3           : YES, END OF TITLE
0345 044C 20E804      PRH2  JSR OUTALL    : PRINT IT
0346 044F E613        INC CHARS
0347 0451 A513        LDA CHARS
0348 0453 4C3B04      JMP PRH0
0349 0456 A513        PRH3  LDA CHARS
0350 0458 C935        PRH4  CMP #$35     : LEAVE EMPTY UP TO 53
0351 045A F005        BEQ PRH5
0352 045C 9003        BCC PRH5
0353 045E 4C6B04      JMP PRH6
0354 0461 209304      PRH5  JSR SPACE     : PRINT A SPACE
0355 0464 E613        INC CHARS
0356 0466 A513        LDA CHARS
0357 0468 4C5B04      JMP PRH4
0358 046B A9C6        PRH6  LDA #(TEXT2   : PRINT "PAGE: "
0359 046D 85F0        STA HULP
0360 046F A903        LDA #(TEXT2
0361 0471 85F1        STA HULP+1
0362 0473 20CC04      JSR MESS0
0363 0476 F8          SED
0364 0477 18          CLC
0365 0478 A515        LDA CURPAG         : INCREMENT CURRENT PAGE
0366 047A 6901        ADC #$01
0367 047C 8515        STA CURPAG
0368 047E AA          TAX
0369 047F A516        LDA CURPAG+1
0370 0481 6900        ADC #$00
0371 0483 8516        STA CURPAG+1
0372 0485 D8          CLD
0373 0486 20F704      JSR WRAX           : AND OUTPUT IT
0374 0489 20F104      JSR CRLF          : CR+LF
0375 048C 209304      JSR SPACE         : SPACE
0376 048F 20F104      JSR CRLF          : CR+LF
0377 0492 60          RTS
0378 0493
0379 0493            : PRINT A SPACE ON A.O.D.
0380 0493
0381 0493 A920        SPACE  LDA #' '
0382 0495 4CE804      JMP OUTALL         : PRINT A SPACE
0383 0498
0384 0498            : SET PRINTER MODE TO WIDE CHARACTERS
0385 0498
0386 0498 A91F        PRWIDE  LDA #$1F
0387 049A 8D1004      MODE   STA PRMODE   : SET PRINTER MODE
0388 049D
0389 049D            : SEND CHARACTER ONLY IF A.O.D. IS PRINTER

```

FORMAT LISTER V1.3

FATE 65XX ASSEMBLER V1.0 PAGE: 0007

```

0390 049D      :
0391 049D 48      OUTPO   PHA
0392 049E AD2105    LDA OUTFLG
0393 04A1 C950      CMP #'P'
0394 04A3 D004      BNE OUTPO1
0395 04A5 68        PLA
0396 04A6 4CE804    JMP OUTALL
0397 04A9 68        OUTPO1 PLA
0398 04AA 60        RTS
0399 04AB      :
0400 04AB      : SET PRINTER MODE TO NORMAL CHARACTERS
0401 04AB      :
0402 04AB A91E      PRNORM  LDA #$1E
0403 04AD D0EB      BNE MODE
0404 04AF      :
0405 04AF      : SEND ESCAPE CHARACTER TO PRINTER
0406 04AF      :
0407 04AF A91B      PRESC   LDA #$1B
0408 04B1 D0EA      BNE OUTPO
0409 04B3      :
0410 04B3      : SET PRINTER TO LONG LINES
0411 04B3      :
0412 04B3 20AF04    PRLONG  JSR PRESC
0413 04B6 A941      LDA #'A'
0414 04B8 D0E3      BNE OUTPO
0415 04BA      :
0416 04BA      : SET PRINTER TO 6 LINES/INCH
0417 04BA      :
0418 04BA 20AF04    PRI6L   JSR PRESC
0419 04BD A936      LDA #'6'
0420 04BF 209D04    JSR OUTPO
0421 04C2 A93B      LDA #$3B
0422 04C4 8502      STA MAXL
0423 04C6 A93E      LDA #$3E
0424 04C8 8D8F05    STA MAXLIN
0425 04CB 60        RTS
0426 04CC      :
0427 04CC      : PRINT MESSAGE (HULP)
0428 04CC      :
0429 04CC A000      MESSO    LDY #$00
0430 04CE B1F0      MESSY    LDA (HULP),Y      : GET MESSAGE CHARACTER
0431 04D0 C93B      CMP #'.'
0432 04D2 F007      BEQ MESSND      : DELIMITER?
0433 04D4 20E804    JSR OUTALL      : PRINT CHARACTER
0434 04D7 C8        INY
0435 04D8 4CCE04    JMP MESSY
0436 04DB 60        MESSND  RTS
0437 04DC      :
0438 04DC      :
0439 04DC      : SENIOR MONITOR ROUTINE VECTORS
0440 04DC      :
0441 04DC 6C00E0     COMIN    JMP ($E000)      : RETURN TO SENIOR MONITOR
0442 04DF 6C02E0     WHEREI   JMP ($E002)      : DETERMINE INPUT DEVICE
0443 04E2 6C04E0     WHEREO   JMP ($E004)      : DETERMINE OUTPUT DEVICE
0444 04E5 6C06E0     INALL    JMP ($E006)      : INPUT FROM A.I.D.
0445 04E8 6C08E0     OUTALL   JMP ($E008)      : OUTPUT TO A.O.D.
0446 04EB 6C0AE0     CLOSEI   JMP ($E00A)      : CLOSE INPUT DEVICE
0447 04EE 6C0CE0     CLOSEO   JMP ($E00C)      : CLOSE OUTPUT DEVICE
0448 04F1 6C0EE0     CRLF     JMP ($E00E)      : CR + LF TO A.O.D.
0449 04F4 6C14E0     RCHEK    JMP ($E014)      : CHECK FOR KEYBOARD INTERRUPT
0450 04F7 6C26E0     WRAX     JMP ($E026)      : BYTES IN A & X TO A.O.D.
0451 04FA 6C58E0     REWIND   JMP ($E058)      : REWIND INPUT DEVICE
0452 04FD 6C5EE0     FORMS    JMP ($E05E)      : FORCE NEW PAGE
0453 0500      :
0454 0500      : .END

```

**** JUNIOR BASIC LISTING ++ DE 6845 GEPROGRAMMEERD ++ ****

```
10 REM---==<<< D E   6 8 4 5   G E P R O G R A M M E E R D >>>==--
12 REM Dit programma is een wijziging en aanvulling
14 REM op het artikel met de zelfde naam in Elektuur okt.'84 .
16 REM
18 REM Aangepast door :
20 REM H.Feijen
22 REM Pr.Clausstraat 30
24 REM 9422 GN SMILDE.
25 REM
26 REM Gemaakt op een Junior met VDU en 16K stat.RAM +
28 REM 16K dyn.RAM en geprint op een Triumph-Adler
30 REM elektronische schrijfmachine omgebouwd tot printer.
32 REM
34 REM Hebt U een printer,zet dan de printer inschakel routine
36 REM in subroutine 3000 en de uitschakel routine in 4000
40 REM
50 INPUT"Gevonden waarden uitprinten ?   (Y/N)";PR$
60 PRINTCHR$(27);:PRINTCHR$(49)
100 REM*****CONSTANTEN*****
105 DIMR(15)
110 R(3)=8
120 K$="REGISTER"
130 L$="MICROSECONDEN"
140 REM***** RO *****
150 PRINT"Het programma is ingesteld op een kristal van 15 Mhz....."
151 PRINT"MOET DE KLOKFREQUENTIE VAN DE VDU KAART AANGEPAST WORDEN ?"
152 PRINT:PRINT:PRINT"<Y/N>":INPUT YN$
153 IFYN$="N"THENR(0)=119:TC=64/120:LPB=8*TC:TSL=120*TC:GOTO300
160 PRINT"HORIZONTALE LIJN LENGTE (CHAR.): "
161 PRINT"voor een 15 MHz kristal is dit 120,voor een 16 MHz 128"
170 INPUT A0
180 R(0)=A0-1
190 TC=64/A0
200 FX=8/TC
210 PRINT"FREQUENTIE = ";FX;" MHz"
220 PRINT"KRISTAL FREQUENTIE (MHz): "
230 INPUTFX
240 TC=1/(FX/8)
250 LPB=R(3)*TC
260 TSL=A0*TC
300 REM***** R1 *****
310 PRINT"AANTAL KARAKTERS PER REGEL:standaard 80"
320 INPUTR(1)
330 DT=R(1)*TC
400 REM***** R2 *****
410 HP=DT+(TSL-1.5*LPB-DT)/2
420 R(2)=HP/TC
500 REM***** R3 *****
600 REM***** R4 *****
610 PRINT"AANTAL LIJNEN PER REGEL:(standaard 9,minimaal 8)"
620 INPUTA
623 IF A<8 THEN PRINT"MINIMAAL 8 LIJNEN !":GOTO610
625 PRINT"AANTAL REGELS OP EEN SCHERM:(standaard 24)"
630 INPUT B
640 TR=(A)*TSL
650 VT=(B+1)*TR
```

**** JUNIOR BASIC LISTING ++ DE 6845 GEPROGRAMMEERD ++ ****

```

660 IF VT<=20000 THEN 680
665 PRINT
670 PRINT"ONMOGELIJK! "
675 PRINT"VERANDER AANTAL LIJNEN OF REGELS A.U.B. "
677 GOTO600
680 Y=INT(20000/TR)
690 R(4)=Y-1
700 REM***** R5 *****:
710 R(5)=INT((20000-Y*TR)/TSL)
800 REM***** R6 *****
810 R(6)=B
815 VD=R(6)*TR
900 REM***** R7 *****
910 R(7)=INT(((TR*Y+TSL*R(5))-(1500+B*TR))/2+B*TR)/TR)
915 VP=R(7)*TR
1000 REM***** R8 *****
1010 R(8)=0
1100 REM***** R9 *****
1110 R(9)=A-1
1120 REM***** R10 *****
1124 PRINT:PRINT:PRINT:PRINT
1126 PRINT"Data in R10 bepaald op welke lijn de cursor start,"
1128 PRINT"of hij stilstaat of knippert, en met welke snelheid."
1130 PRINT:PRINT:PRINT
1200 PRINT:PRINT
1202 GOSUB3000
1204 PRINT"CURSOR MODE:          STIL          GEEN          SNEL          LANGZAAM"
1205 PRINT"===== "
1206 PRINT"START OP LIJN: 0          0          32          64          96"
1207 PRINT" - - - 1          1          33          65          97"
1208 PRINT" - - - 2          2          34          66          98"
1209 PRINT" - - - 3          3          35          67          99"
1210 PRINT" - - - 4          4          36          68          100"
1211 PRINT" - - - 5          5          37          69          101"
1212 PRINT" - - - 6          6          38          70          102"
1213 PRINT" - - - 7          7          39          71          103"
1214 PRINT" - - - 8          8          40          72          104"
1215 PRINT:PRINT:
1218 GOSUB4000
1220 PRINT:PRINT:INPUT "GEEF WAARDE R10";R(10)
1225 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
1231 REM***** R11 *****
1232 PRINT"Data in R11 bepaald op welke lijn de cursor eindigt."
1234 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
1235 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
1240 PRINT"EINDIGEN OP LIJN: 0    DATA = 0"
1245 PRINT:PRINT:PRINT
1250 PRINT"EINDIGEN OP LIJN: 8    DATA = 8"
1290 PRINT:PRINT:INPUT "GEEF WAARDE R11";R(11)
1300 REM***** R12,R13,R14 & R15 *****
1310 R(12)=0
1320 R(13)=0
1330 R(14)=0
1340 R(15)=0
1350 PRINT:PRINT
1351 GOSUB3000

```

**** JUNIOR BASIC LISTING ++ DE 6845 GEPROGRAMMEERD ++ ****

```

1352 PRINT"SCHEM FORMAAT = ";R(1);" * ";B
1354 PRINT:PRINT
1700 FOR Q = 0 TO 15
1710 PRINTK$;" R";Q;
1720 PRINT TAB(20);" = ";
1727 Z2=R(Q)
1730 GOSUB 2000
1740 PRINT"          (";R(Q);"decimaal.)"
1741 POKE6722+Q,R(Q)
1750 NEXT Q
1755 PRINT:PRINT:PRINT
1757 GOSUB4000
1760 INPUT"Voor vervolg toets <SPATIE - RETURN>";VV$
1770 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
1780 GOSUB3000
1800 PRINT" KLOK PERIODE TIJD                ";TC;L$
1810 PRINT" LIJN SYNC.PULS WIJDTE           ";LPB;L$
1815 PRINT" LIJN SYNC.PULS PERIODE TIJD      ";TSL;L$
1830 PRINT" HORIZONTALE DIPLAY TIJD          ";DT;L$
1840 PRINT" HORIZONTALE POSITIE              ";HP;L$
1850 PRINT" KARAKTER LIJN PERIODE TIJD       ";TR;L$
1855 VE=Y*TR+R(5)*TSL
1860 PRINT" RASTER SYNC.PERIODE TIJD        ";VE;L$
1865 PRINT" VERTIKALE DISPLAY TIJD          ";VD;L$
1867 PRINT" VERTIKALE POSITIE               ";VP;L$
1868 GOSUB4000
1869 PRINT:PRINT:PRINT
1870 PRINT"CRTC CONTROLLER VOLGENS DEZE BEREKENING VERANDEREN ?"
1871 PRINT:PRINT:PRINT"<Y/N>":INPUT NY$
1872 IF NY$="Y" THEN 1874
1873 GOTO 1930
1874 REM***** MASTER RESET VDU op HEX 0F9E *****
1877 POKE8256,158:POKE8257,15:X=USR(X)
1878 REM
1900 PRINT"Nog iets veranderen aan cursor ?"
1910 PRINT:PRINT:PRINT:PRINT:INPUT"Tik Y/N....<RETURN>";YN$
1920 IF YN$="Y" THEN PRINTCHR$(27);:PRINTCHR$(49):GOTO 1126
1930 PRINT:PRINT:PRINT"E I N D E":END
2000 REM***** DEC TO HEX *****
2010 PRINT"$";
2020 FOR Z=1 TO 0 STEP -1
2030 Z1=INT(Z2/16^Z)
2040 Z2=Z2-Z1*16^Z
2050 Z1=Z1+48
2060 IF Z1>57 THEN Z1=Z1+7
2070 PRINT CHR$(Z1);
2080 NEXT Z:RETURN
3000 IF (PR$<"Y") OR (PR$>"Y") THEN RETURN
3010 POKE6744,128:RETURN
4000 POKE6744,00:RETURN

```

>

INSERT/DELETE COMMANDS

PROTON 650X ASSEMBLER V4.4 PAGE: 0001

```

0001 0000      .TIT 'INSERT/DELETE COMMANDS'
0002 0000      .OPT GEN
0003 0000
0004 0000      ;
0005 0000      ; *****
0006 0000      ; *
0007 0000      ; * INSERT - DELETE CHARACTER ROUTINES T.B.V. DE VDU-KAART *
0008 0000      ; *
0009 0000      ; *****
0010 0000      ;
0011 0000      ; AUTEUR: F.J.M. SMEEHUIJZEN
0012 0000      ; LIPPEDAL 19
0013 0000      ; 2904 CL CAPELLE AAN DEN IJSSEL
0014 0000      ; TEL: 010-512507
0015 0000      ;
0016 0000      ; SINDE ENIGE TIJD WERK IK MET DE CPU/VDU KAARTEN VAN
0017 0000      ; ELEKTUUR IN KOMBINATIE MET DE PROTON MONITOR.
0018 0000      ; HIERBIJ MAAK IK TEVENS GEBRUIK VAN DE PROTON EDITOR.
0019 0000      ; DEZE EDITOR IS VOORZIEN VAN EEN CHARACTER DELETE-
0020 0000      ; EN EEN CHARACTER-INSERT ROUTINE.
0021 0000      ; OM NU OOK VIA HET BEELDSCHERM VAN DEZE ROUTINES GEBRUIK
0022 0000      ; TE KUNNEN MAKEN, HEB IK DE NAVOLGENDE ROUTINES ONTWIKKELD.
0023 0000      ;
0024 0000      ; DELETE CHARACTER ROUTINE.
0025 0000      ; -----
0026 0000      ;
0027 0000      ; DE DELETE-ROUTINE WORDT GEAKTIVEERD DOOR EEN
0028 0000      ; CONTROL-D(DELETE) IN TE TOETSEN.
0029 0000      ; HIERBIJ WORDT HET TEKEN OP DE PLAATS VAN DE
0030 0000      ; CURSOR VERWIJDERD EN DE REST VAN DE REGEL
0031 0000      ; NAAR LINKS OPGESCHOVEN.
0032 0000      ;
0033 0000      ; INSERT CHARACTER ROUTINE.
0034 0000      ; -----
0035 0000      ;
0036 0000      ; DE INSERT-ROUTINE WORDT GEAKTIVEERD DOOR EEN
0037 0000      ; CONTROL-I(INSERT) IN TE TOETSEN.
0038 0000      ; HIERBIJ WORDT DE TEKST VANAF DE CURSOR EEN PLAATS
0039 0000      ; NAAR RECHTS OPGESCHOVEN EN OP DE PLAATS VAN DE
0040 0000      ; CURSOR EEN SPATIE GEZET, WAARNA EEN GEWENST TEKEN
0041 0000      ; KAN WORDEN INGETOETST.
0042 0000      ;
0043 0000      ; OM BEIDE ROUTINES TE KUNNEN GEBRUIKEN DIENT ALLEREERST
0044 0000      ; DE COMMAND TABLE 'COTABX' TE WORDEN AANGEVULD EN
0045 0000      ; VERVOLGENS DE COMMAND ADDRESS TABLE TE WORDEN
0046 0000      ; UITGEBREID.
0047 0000      ;
0048 0000      ; *** ZERO PAGE POINTERS ***
0049 0000      ;
0050 0000      ;      *=$0000
0051 0000      ;
0052 0000      ; RAMPTR      *==+6      ; RAMPONTER
0053 0000      ; OFFSET      *==+4      ; AMOUNT OF CHARACTERS TO EOL
0054 0000      ; CLN          *==+4      ; CURRENT LINE POINTER
0055 0000      ; SCRPTR      *==+4      ; SLAVE SCREEN POINTER
0056 0000      ; COL          *==+2      ; CURRENT COLUMN
0057 0000      ; TEMCOL      *==+2      ; SLAVE COLUMN
0058 0000      ; CHAPLN      *==+1      ; CHARACTERS PER LINE
0059 0000      ;
0060 0000      ; *** VDU-ROUTINES ***
0061 0000      ;
0062 0000      ; VIDEND      = $F9EB
0063 0000      ; CRAMPT      = $FB8D
0064 0000      ;
0065 0000      ;      *=$0200
0066 0000      ;
0067 0000      ;
0068 0000      ; 204402 DELETE JSR CMPADR      ; CURRENT CURSOR POSITION
0069 0000      ; 18      CLC
0070 0000      ; A516      LDA CHAPLN      ; COMPUTE AMOUNT OF
0071 0000      ; E514      SBC TEMCOL      ; CHARACTERS TO BE
0072 0000      ; 8506      STA OFFSET      ; SHIFTED TO THE LEFT
0073 0000      ; A001      LDY #01
0074 0000      ; B100      LDA (RAMPTR),Y      ; MOVE CHARACTERS
0075 0000      ; 88      DEY      ; BEHIND CURSOR
0076 0000      ; 9100      STA (RAMPTR),Y      ; ONE POSITION TO

```

PROTON 650X ASSEMBLER V4.4 PAGE: 0002

ERRORS: 0000

(000)

BOEKEN:

Lien, D.A.

Het Basic-Handboek.

Het meest noodzakel

gereedschap voor de microcom- x
puter bezitter: een encyclo- o
pedie met meer dan 500 BASIC- x
woorden verklaard en toege- o
licht met testprogramma's. x
Op onze redactie is men van o
mening dat het nuttig is bij x
uitpluizen van Basic-program- o
ma's van andere computers. x

Fabelachtig printen in kleur of zwart/wit



OKIMATE 20



OKI MICROLINE 182



Futuristische OKI-afdrukkers erkend de beste in Nederland. Nu ook voor iedereen betaalbaar geworden. Aansluitbaar op Commodore 64, BBC, MSX e.a. homecomputers.

OKImate 20 v.a. fl. 985.--
Microline 182 v.a. fl. 1170.--
excl. b.t.w. Bruto adviesprijs.

OKI printers, door computers voor computers.

Vraag de dealerlijst bij de officiële importeur

Technitron B.V.

Zwarteweg 110, Postbus 14, 1430 AA Aalsmeer

tel. 02977-22456 - telefax 02977-40968 - telex 13301